CANADIAN THESES

THÈSES CANADIENNES

DANG, TRUNG

VECTORIZATION OF THE HEATER LASER ...

ISBN

0-315-69921-3

National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

NL-339 (r. 88/04) c

Canada

UNIVERSITY OF ALBERTA

VECTORIZATION OF THE HEATER LASER PROPAGATION CODE

BY

TRUNG DANG

A THESIS

Submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science.**

DEPARTMENT OF ELECTRICAL ENGINEERING

EDMONTON, ALBERTA

Fall 1991

National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-69921-3

Canada

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: **TRUNG DANG**

TITLE OF THESIS: **VECTORIZATION OF THE HEATER LASER PROPAGATION CODE**

DEGREE: **MASTER OF SCIENCE**

YEAR THIS DEGREE GRANTED: **1991**

Permission is hereby granted to the University of Alberta library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purpose only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

#11 10650-110 Street
Edmonton, Alberta
T5H-3C8, CANADA

September 12th, 1991

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Vectorization of the HEATER Laser Propagation Code**, submitted by Trung Dang in partial fulfillment of the requirements for the degree of *Master of Science* in *laser-plasma interaction.*

Supervisor: Dr. C.E. Capjack
(Associate Dean, Engineering)

Supervisor: Dr. C.R. James
(Vice President, Research)

Committee member: Dr. J.C. Samson
(Professor, Physics)

Committee member: Dr. F.E. Vermeulen
(Professor, Electrical Engineering)

Date: Aug. 28th, 1991

Hiểu được gía trị tương đối của khoa học đã khó!
Hiểu được chân lý tuyệt đối của thượng đế càng khó hơn!

Kính dâng hương hồn ông nội.
Kính dâng thầy mẹ.
Thương tặng vợ hiền và hai con Bảo, Nhi.

ABSTRACT

An understanding of the interaction of laser light with matter is of importance both in industrial applications and in fundamental applied research. The subject of the present investigation is the application of lasers as drivers in inertially confined fusion and as pumping agents in the production of x-ray lasers. Specifically, the current research is devoted to restructuring of the HEATER subroutine, which simulates the laser propagation in an underdense plasma region. The HEATER subroutine was found to consume a large fraction of the CPU time (60 to 70 percent) when running the CASTOR plasma simulation code. The program is modified in order to take advantage of speed gains possible through vectorization when running on the CRAY XMP and CYBER 205 vector computers. The modified HEATER routine also ensures that the work done by ponderomotive forces is accounted for in a self-consistent manner so that the conservation of energy is preserved. The laser beam propagation package is described assuming a 2-D cylindrical geometry and is structured so that it can be incorporated into any Eulerian or Lagrangian hydrodynamic code.

ACKNOWLEDGEMENT

I would like to express my deep and sincere gratitude to Dr. C. E. Capjack and Dr. C. R. James, my supervisors, for setting the direction of my research and for providing continuous guidance and motivation throughout the last three years. I sincerely thank both of them for their patience, understanding, intellectual and financial support.

I would sincerely thank Dr. R. Rankin who has helped me extensively to clarify and understand about my project. I would also thank Dr. M. R. Amin for his useful discussion. I also would like to extend my appreciation to Dr. F. E. Vermeulen and Dr J. C. Samson, members of a examining committee, for their invaluable time spent on this thesis. I also would like to thank the Electrical Department which has given me a opportunity to complete my study and provided me with financial support through T.A. work.

Finally, the forbearance, sacrifices and continuous encouragement from my wife during the period of this research, the emotional support from parents and relatives are very appreciated. I also would like to extend my thanks to Mr. Kim Vu who has spent time to proofread my thesis.

TABLE OF CONTENTS

CHAPTER		PAGE
1.	**INTRODUCTION**	1
	1.1 The laser: a driver for nuclear fusion	1
	1.2 Significance of the problems	2
	1.3 Past work review and survey	4
	1.3.1 Nature of beam propagation	4
	1.3.2 Numerical approach to beam propagation	5
	1. Ray tracing approach	5
	2. Electromagnetic equation approach	6
	1.4 Objectives and organization of the thesis	6
2.	**VECTORIZATION OF A FORTRAN PROGRAM**	9
	2.1 Preliminary remarks	9
	2.2 Brief review of vector computer architecture	9
	2.3 Vectorization	11
	2.4 Strategy for vectorization	12
	2.5 Vectorization inhibitors	14
	2.6 Vectorization on the CRAY XMP	15
	2.6.1 Hardware dependency	15
	2.6.2 Software dependency: the compiler	16
	1. Avoiding calling subroutine and external functions within a loop	17
	2. Increasing arithmetic chaining	18
	3. Combining short loops and short subroutines	20
	4. Optimizing the innermost DO loops	21

5. Unrolling short DO loops ——— 22
6. Eliminating unneeded storage ——— 23
7. Eliminating common sub-expressions ——— 24
8. Avoiding recurrence ——— 24
9. Restructuring IF statements ——— 26
10. Avoiding I/O statements ——— 27
11. Avoiding the use of assigned GO TO statements 28
2.7 Vectorization on the CYBER 205 ——— 29
1. Link triad ——— 30
2. Gather/Scatter instructions ——— 31
3. Problems on multi-dimensional arrays ——— 32
2.8 Vectorization summary ——— 33
3. LASER PROPAGATION MODEL:
THEORY AND NUMERICAL SOLUTION 35
3.1 Theory and Equations ——— 35
3.1.1 The wave equation ——— 35
3.1.2 The WKB approximation to a wave equation 36
3.1.3 The ponderomotive force equations ——— 40
3.1.4 The energy absorption equation ——— 43
3.2 Numerical solution ——— 44
3.2.1 Radial discretization ——— 44
3.2.2 Beam advection in axial direction ——— 46
4. VECTORIZATION OF HEATER SUBROUTINE 48
4.1 Structure of the HEATER code ——— 48
4.1.1 Laser off ——— 49
4.1.2 Laser on ——— 49

4.1.3 Input power — 51
1. Compression pulse — 51
2. Gaussian pulse — 51
3. Triple ramp pulse — 52
4. Plane wave with Gaussian perturbation pulse — 52
4.1.4 Absorption coefficients — 53
4.1.5 Entrance plasma conditions — 53
4.1.6 Solution at all radial and axial meshes — 54
4.1.7 Normalization of volume elements — 56
4.2 Implementation of HEATER on the CRAY XMP — 57
4.2.1 Avoiding assigned GO TO statements — 57
4.2.2 Converting scalar variables into vector arrays — 59
4.2.3 Unrolling short DO loops — 59
4.2.4 Referencing external functions within a loop — 60
4.2.5 Avoiding repeated calculations — 60
4.2.6 Increasing chaining processes — 63
4.2.7 Using a new vectorizable algorithm — 64
4.3 Implementation of HEATER on the CYBER 205 — 66
4.4 Benchmarking for the CRAY XMP and CYBER 205 — 67
5. COMPUTATIONAL RESULTS — 73
5.1 Preliminary remark for MHD code: CASTOR — 73
5.1.1 Physical description — 73
5.1.2 Restriction on the complexity — 74
5.1.3 Programming control and utilities — 74
5.1.4 Relationship between HEATER and CASTOR — 75
5.2 Simulation example — 76

5.3 Results: three-dimensional representation ——— 79
5.3.1 Electron temperature profiles ——— 79
5.3.2 Ion temperature profiles ——— 80
5.3.3 Absorption profiles ——— 80
5.3.4 Ponderomotive force profiles ——— 80
5.3.5 Electron density profiles ——— 82
5.3.6 Laser flux profiles ——— 83
5.3.7 Laser and plasma profiles without ponderomotive forces ——— 84
5.4 Results: two-dimensional representation ——— 85
5.4.1 Laser intensity profiles ——— 85
5.4.2 Ponderomotive force profiles ——— 86
5.4.3 Electron density profiles ——— 86
5.5 Self-focusing and filamentation of laser light ——— 87
5.5.1 Physical effects on self-focusing and filamentation processes ——— 87
5.5.2 Applications and effects of filamentation ——— 88
6. CONCLUSION 103
BIBLIOGRAPHY 107
APPENDIXES 111
TABLES 128
FLOW CHARTS 130
HEATER SUBROUTINE LISTING 150
CYCLIC SUBROUTINE LISTING 178

CHAPTER ONE
INTRODUCTION

1.1 THE LASER: A DRIVER FOR NUCLEAR FUSION

The use of high power lasers is one of the strategies being pursued in nuclear fusion research. Several techniques have been proposed for delivering energy to fusion targets by using a laser [3,4]. In laser-driven inertial fusion, energy is obtained from thermonuclear micro-explosions induced by the irradiation of a small quantity of fusionable material by a very powerful laser pulse [3,5]. In a proposed reactor design, a small hollow, spherical target is irradiated by converging beams with very high power intensity in a very short duration. The plasma created by the ablation of the outer layer of the target produces a reaction force directed toward the center of the target. As the shell collapses, the fuel is compressed to a very high density and at the same time heated to a very high temperature. As a result, the conditions for thermonuclear fusion can be achieved.

Presently, there are other proposals involving the use of ion beams as a driver to achieve inertial confinement fusion (ICF) [30]. One approach is to use light ions (eg. protons) accelerated to a very high energy (MeV). This research is being developed at Sandia Laboratories in the United States. However, major problems involving beam focusing and particle interaction physics still remain to be solved in a reactor context. An alternative proposal is to use heavy ions (eg. Uranium) of high energy (GeV) to drive ICF. The interaction

physics, beam generation and focusing of this latter approach are expected to be well understood in a near future. However, the large initial capital cost of this experiments has limited study to only theoretical analysis at present.

Separate from fusion applications, laser heated plasmas can also be used in X-ray lasers [29,31-34]. A laser heated plasma column can serve as a gain medium for producing very short coherent wavelength sources that are 100 times shorter than conventional laser wavelengths.

Before any of the above laser schemes can lead to useful commercial applications, the physics of the laser-plasma interaction process must be better understood. This is being approached through experiments and through the use of theoretical and numerical techniques. Today, with the availability of powerful high-speed computers, the simulation of laser-plasma interaction processes has become feasible.

1.2 SIGNIFICANCE OF THE PROBLEMS

When a high power laser beam strikes a target with sufficient intensity, an expanding plasma is produced. The process involves most of the laser light being absorbed through inverse bremsstrahlung [5-27] in a region with a large spatial density gradient. The energy absorbed by the electrons through inverse bremsstrahlung absorption is then collisionally transferred to the ions. The maximum plasma density accessed by the laser beam is defined by the critical density N_c, where the laser frequency is equal to the plasma frequency. The plasma

density typically drops to a fraction of a percent of N_c within a few hundred μm from the solid target interface [8]. The plasma tends to be pushed away from regions in which the beam intensity is locally higher through the combination of ponderomotive and thermal forces. This feature leads to self focusing of the laser beam as the laser propagates through the plasma density gradient [8-11,14,15,17,19-22]. The laser energy that is absorbed through inverse bremstrahlung further heats the plasma and this enhances the change in the plasma density profile. This disturbance in the plasma causes the laser light to both refract and diffract as it propagates in the plasma [16,20,21,25].

It has been shown by Emery et al. [6] that the laser-driven targets may generate the shedding of strong vortex structures from the ablation layer whenever the density and pressure profile become non-collinear. The conditions may be produced by perturbations in the target profile itself or by asymmetry in the laser profile. The vortex shedding process creates a vortex street which extends from the ablation layer to the underdense region. This vortex shedding may have serious implications with respect to the laser absorption process, self-focusing plasma instabilities and turbulence in the underdense region.

The challenge facing the ICF is the creation of spatially uniform pressure distribution within the fuel target. Therefore, the direct laser-driven ICF must require an intense pulse of long duration, and a smooth intensity profile.

In order to better understand the laser-driven inertial fusion

process, it is essential to study the propagation of a laser beam in a plasma. The research undertaken in this thesis will focus on the mechanism of laser beam propagation in an underdense plasma region. A literature survey of numerical simulation techniques that have been used for describing laser beam propagation in plasmas will be discussed first.

1.3 PAST WORK REVIEW AND SURVEY

1.3.1 Nature of beam propagation.

Beam propagation plays an important role in the laser-plasma interaction process. Self-focusing and defocusing are normal characteristics of beam propagation in plasma.

Steinhauer and Ahlstrom [28] have shown that with a parabolic density profile, the laser beam was trapped and followed a pattern of focusing and defocusing. Trapping of the laser beam has been shown to occur when the radial density profile has a minimum value along the axis of propagation. Feit and Fleck [7,10,11] have treated laser beam propagation and plasma heating self-consistently. The expansion of the plasma due to laser heating creates locally low densities on the axis of propagation. As a result, the beam trapping is observed during this process.

McMullin et al. [24] used the normal mode theory to derive the oscillating Gaussian solution for a laser beam in a plasma waveguide with a quadratic index of refraction. This showed that, with a radial density profile having a minimum value along its axis, beam aberration occurs even after traveling a short distance along the plasma column.

1.3.2 Numerical approach to beam propagation.

There are various approaches that may be used to simulate laser beam propagation in a plasma medium. One technique is to use the ray-tracing method which treats the laser beam as one or a bundle of light rays propagating through the media [13-18]. An alternative approach is to numerically solve the wave equation. The latter approach accounts for the wave characteristics of the beam [8,9,19-22]. Both approaches include physical effects such as diffraction, refraction, absorption and ponderomotive forces. Ray tracing has intrinsic advantages for describing hydrodynamic implosion calculations because beam propagation can be described near the critical density N_c; however, the filamentation process in subcritical density plasmas is more accurately described by the paraxial wave approach [22].

1. RAY-TRACING APPROACH

One technique for analyzing laser beam propagation is the use of a self-consistent ray tracing model. Such simulations have been used to explain the dependence of thermal self-focusing on beam size, plasma scale length, laser wave length and beam intensity [15,17]. An example of the ray tracing model is the two-dimensional Lagrangian LASNEX code [14,17]. This program has been used to study many important physical processes in laser-driven fusion. In LASNEX, the laser is simulated by using one or more bundles of rays. A limitation of this model is that information about electric or magnetic fields of the incident radiation is not accounted for.

A ray tracing model is also used in SAGE [15], a two-dimensional

Eulerian hydrodynamic code which includes only the thermal kinetic forces. A limitation of the laser beam propagation model in SAGE is that the ponderomotive forces are not accounted for. These forces can become very important in the region near the critical density.

2. ELECTROMAGNETIC EQUATION APPROACH

Another approach for describing electromagnetic beam propagation in a plasma is to solve the wave equation [8,9,19-22]. An analytical or numerical solution to the wave equation is more complete than the ray tracing model because the wave nature of the laser beam is accounted for. In most cases, the paraxial approximation to the wave equation is assumed.

F. Lasinski et al. [28] have used a Gaussian ansatz to solve the wave equation. S. Sartang et al. [9] and D. J. Nicholas, S. G. Sajjadi [8] have used the WKB method to approximate the wave equation. Differencing schemes such as the Crank-Nicolson or the ADI (alternative difference implicit) [35] have been used to numerically solve the wave equation. The use of finite difference in solving a wave equation usually results in a need to solve tridiagonal matrices, which is usually accomplished by using a Gaussian elimination method. Laser absorption, ponderomotive forces and diffraction effects are included. A Gaussian radial beam profile is assumed in most of these studies.

1.4 OBJECTIVES AND ORGANIZATION OF THE THESIS

The purpose of the present work is to investigate the propagation

of a laser beam in an underdense plasma region. The propagation of the beam is described by solving the wave equation. The numerical solution of the wave equations results in the need to invert tridiagonal matrices. The Cyclic Reduction method will be used in order to take advantage of speed gains possible through vectorization. All parts of the propagation routine amendable to vectorization will be vectorized so that the routine can be ported to the CRAY XMP and CYBER 205 vector computers. The ponderomotive forces are calculated consistently in both the axial and radial directions. An appropriate scaling factor to the beam intensity will be applied so that the work done by the ponderomotive forces can be accounted for in a consistent manner. In particular, energy balance is ensured throughout the whole simulation process. The new version of the laser propagation routine (HEATER) can be ported into any two dimensional (r,z) Eulerian or Lagrangian hydrodynamic code to describe laser beam propagation.

This thesis consists of six chapters. In Chapter 1, an introduction to the relevance of the proposed research is formulated. This is followed by a survey of past research that has been reported on beam propagation modeling (ray tracing and numerically solving a wave equation approach). In Chapter 2, the vectorization of FORTRAN programs is discussed. The strategy of vectorization including inhibitors are discussed for both the register-to-register (CRAY XMP) and memory-to-memory (the CYBER 205) vector computers. A summary of vectorization techniques is outlined. In Chapter 3, both the theory of, and numerical solution to the beam propagation equation are described. In the theory section, the paraxial approximation to the

wave equation is derived, followed by the derivation of ponderomotive forces and laser absorption equations. In the numerical section, the technique that is used to solve the paraxial equation is described. This includes a cubic spline approximation to the radial beam profile in order to ensure accuracy, flexibility and continuity of beam energy. The implicit Crank-Nicolson technique is applied to solve for beam advection in the axial direction. The structure and vectorization of the HEATER subroutine are discussed in Chapter 4. In that chapter, HEATER's structure is subdivided into number of logical blocks. The function of each block is explained briefly in chronological order. The implementation of HEATER subroutine on the CYBER 205 and the CRAY XMP is also examined. New vectorizable algorithms are used in HEATER so that the performance can be improved when running on vector computers. In Chapter 5, an example is given to simulate laser-propagation through a preformed plasma. The parameters used are similar to those employed in [19]. HEATER is interfaced with the hydrodynamic code CASTOR to study the laser-plasma interaction processes. Processes such as ponderomotive work, filamentation, absorption, refraction, diffraction and beam focusing are illustrated using 2D and 3D plots. Chapter 6 summarizes the results obtained from vectorization. It includes the author's suggestions and comments for further improvements.

In this research, SI units are used with the exceptions that the temperature is in eV and power is in Watts.

CHAPTER TWO
VECTORIZATION OF A FORTRAN PROGRAM

2.1 PRELIMINARY REMARK

The solution of many large scientific and engineering problems is currently best addressed by using powerful and high-performance computers such as the CYBER-205, CRAY-1, CRAY-XMP, CRAY-YMP, FUJITSU VP-400, ETA-10 and CONVEX supercomputers. These computers share a special type of architecture known as vector and/or parallel processing which distinguish them from scalar computers. This feature greatly improves performance, which is otherwise unattainable with single processor scalar computers. In order to take advantage in performance increase, programs and numerical algorithms have to be adapted or even restructured to take advantage of this special architecture.

2.2 BRIEF REVIEW OF VECTOR COMPUTER ARCHITECTURE

Most of the current supercomputers have a small scale parallel architecture with several powerful Central Processing Units (CPUs) coupled with a large shared memory. Each CPU consists of a scalar and a vector processor. The vector module is able to perform multiple operations per instruction in a pipeline fashion. The pipeline concept is a streamlining of the conventional scalar processor, a recognition that the most intense use of a computer is almost always in a loop, doing the same operations to many different operands. In other words,

pipelining concepts in vector processor allows more than one instructions to execute at different stages at the same time. Instruction pipelining is achieved by dividing the execution into a number of phases such as fetch, decode, operant fetch and instruction execution. Pipelined units which can function independently of each other include:

- Vector added / subtracted pipes that perform addition / subtraction instructions.
- Vector multiplied pipes that execute floating point or integer multiplication or division instructions.
- Vector loaded pipes that move data from memory into vector registers or memory.
- Vector stored pipes that move data to memory from vector registers or memory.

Each of these pipes may be in the execution phase at the same time. For example, the vector processor may simultaneously perform a vector addition A + B, a vector multiplication C * D and a vector LOAD and STORE. The functional units of the vector processor module often chain operations together automatically. Vector operations can be performed between vector registers only, or between themselves and scalar registers. Besides scalar and vector registers, the Mask registers are used to impose control on vector operations.

A detailed description of specific supercomputers architecture can be found in [36-39,45,46]. Each type may have unique characteristics. The architecture in both the CYBER 205 and CRAY XMP has been previously discussed. These include vector pipelines such as

vector registers, scalar registers, address registers, and temporary storage registers. One goal of this project is to be able to take advantage of the speed gains possible through vectorization on both the CYBER 205 and CRAY XMP supercomputer. The similarities and differences between these two computer types may be briefly summarized as follows:

(i) The CYBER 205 vector computer pipelines communicate directly with the main memory, thus representing so called a memory-to-memory machine.

(ii) The CRAY XMP vector computer is provided with fast registers (vector registers) which act as a high-speed memory to feed the functional units and to communicate with the main memory. This vector computer is called a register-to-register machine.

2.3 VECTORIZATION

Vectorization enables a computer to perform calculations with vector rather than scalar instructions. The compilers that are provided with these machines are able to generate vector machine codes. In this project, implicit vectorization is used so that the source programs are written in standard FORTRAN 77, from which the compiler is able to generate the appropriate vector instructions. The advantage of implicit vectorization is portability. Since the program is written in standard FORTRAN 77, it can be run on any type of computer ranging from micro computers to supercomputers. However, the drawback is that the program may not take full advantage of vectorization when running on the supercomputers and may sometimes run

even slower than programs whose code is specified explicitly. Explicit vectorization means that programs are coded specifically according to each computer architecture design. For example, the CFT77 [45] (a CRAY version of FORTRAN 77) and FT-200 [46] (a CYBER version of FORTRAN 77) are used to vectorize explicitly. Both of them are compatible to the standard version, but extensions to special routines and explicit statements are additional features which depend on hardware structures. The advantage of explicit vectorization is that programs can be restructured to take full benefit of vectorization (eg. linear and non-linear recurrence). However, a major drawback is that a program with explicit declarations becomes dependent upon a specific computer architecture. Program portions may not conform to ANSI FORTRAN standards, and therefore, program portability is lost.

2.4 STRATEGY FOR VECTORIZATION

The production codes in the field of computational physics (ie. HEATER, CASTOR) are usually large and complex. Vectorization of such a code appears to be difficult and challenging in the beginning. Therefore, some strategies must be planned to ease the vectorization task of such codes.

As a first step, a programmer may generate a FLOW-TRACE to identify routines which are most frequently used. A FLOW-TRACE is a run time statistic which appears in tabular form in a summary after the execution of the program. By using this summary, a programmer can determine which program segments or routines needed to be considered for vectorizing or restructuring. For instance, a subroutine, which

consumes 80% of executable time, must receive a higher priority for restructuring (vectorizing) than a subroutine that uses only 2% of the total execution time.

As a second step, a programmer can compare the scalar and vector execution of a program to determine the degree of optimization already being performed by the compiler. This can be done by making a normal run of the program in 'scalar' mode and in 'vector' mode. Most compilers provide a simple mechanism in the control statement to accomplish this by forcing the compiler to turn vectorization on (vector mode) or off (scalar mode) [45,46]. This will generate the complete output list of the vectorized and non-vectorized loops together with the reasons for non-vectorizability. Possibilities that may exist when comparing vector and scalar modes include:

(i) If the vector and scalar execution in subroutines or DO loops have identical times, the compiler has not been able to optimize these particular DO loops or subroutines. Therefore, a large performance improvement can be expected if they are restructured.

(ii) If the vector execution time of subroutines or DO loops is significantly (factor of 5 to 10 times) less than the scalar execution time, further restructuring may be unnecessary.

In the case of a highly serial algorithm involving linear and non-linear recurrences, step (i) and step (ii) may not indicate hints of how DO loops or subroutines should be restructured. One option is to replace these codes with vectorized algorithms. As an example, in the HEATER code, the Cyclic Reduction algorithm [47] for solving tridiagonal matrices is used instead of a standard Gaussian

elimination algorithm. An alternative option is to use explicit vectorization and/or special calling subroutines (which are designed differently for each type of supercomputer) to overcome difficulties such as linear and non-linear recurrence. Details of explicit vectorization can vary from system to system [45,46] depending on each hardware structure and compilers.

2.5 VECTORIZATION INHIBITORS

Most compilers on supercomputers only vectorize the innermost DO loops. The following inhibitors are outlined to discuss the optimization on both the CRAY XMP and CYBER 205. Inhibitors which prevent a DO loop from being vectorized include:

(i) Calling SUBROUTINE and/or referencing to external functions (usually user defined functions).

(ii) Assigning any I/O statements.

(iii) Assigning GO TO statements.

(iv) Using IF statements within DO loops. For the CRAY XMP computers, IF statements are vectorizable. However, the level of vectorization decreases as the level of nesting increases.

(v) Referencing to character variables.

(vi) Recursion in fetching and storing of the same array.

(vii) Backward transferring of data.

Each of inhibitors described above is examined in detail for both the CRAY XMP and CYBER 205 computers. Vectorization techniques for each type of computer are also described.

2.6 VECTORIZATION ON THE CRAY XMP

2.6.1 HARDWARE DEPENDENCY

The CRAY XMP is a register-to-register vector computer. Knowledge of hardware structure and principal operation of such a machine is necessary for optimally vectorizing program source codes. Detailed hardware structures of this machine can be found in [36-39,45]. There are some basic features that may be included. This machine moves data from memory to vector registers and performs computations with vector-register operands. The results are computed and retained in the registers for further use and then stored into memory. The CRAY XMP has eight vector registers, each having 64 elements that are 64 bits wide. When a loop of arbitrary length is vectorized, it is done in vector strips of length 64. There is a start-up time to fetch data from memory, which depends on the vector length. The registers must be emptied again before a new 64 element segment can be fed into the active vector register. For example, in vector operation of the form (A+B)*C, the result register of the first operation (the addition) is used as the input register for the next operation (the multiplication). The execution of the second operation begins as soon as the first element of the result vector arrives in the result register. This mode of operation is called the 'chaining' process. It saves the access time of storing the intermediate result A+B into the memory.

A programmer wanting optimal performance should try to reuse intermediate results as often as possible, filling and emptying vector registers while, if possible, useful work is performed in the

functional units.

There are some basic characteristics that affect the performance of a FORTRAN program on this type of machine. First, the performance usually varies inversely with the clock cycle time. The shorter the clock cycle, the faster the operation. Secondly, the instruction issue rate also affects the performance. For example, one instruction may require one or two clock cycles for execution. Thirdly, the size and number of vector registers also play an important role in determining performance. This feature allows intermediate results to be saved temporarily and reused during the next operation. If the number of vector registers increases, the number of chaining operations increases. The memory size also affects the speed of computation because the data transfer rate from memory to memory (or register) is faster than the transfer rate from the I/O. The CRAY XMP can concurrently fetch two vectors, perform the addition instruction and then store the result back into memory. As a result, if the number of memory paths is increased, then the performance is also improved.

2.6.2 SOFTWARE DEPENDENCY: THE COMPILER

A compiler is a computer program that reads the program source code as data and translates it into machine code that can be read and interpreted as instructions and data understandable at the computer's hardware level. There are several revised versions of FORTRAN 77 which are specifically designed for each type of computer. For example, CFT77 is a CRAY FORTRAN compiler designed specifically for the CRAY XMP [45]. FT-200 is a revised version of FORTRAN 77, which works best on the CYBER 205 [46]. These compilers are also compatible with the

standard ANSI-FORTRAN.

In the above compilers, additional routines and special functions have been added in order to optimize performance for each type of computer. Some compilers are "smart enough" to automatically optimize the source program by performing steps such as reducing instruction sets, moving invariant codes out of DO loops, evaluating constant expressions, instruction shuffling, and register scheduling. However, most vectorizing compilers are inhibited from performing these techniques because their analysis is blocked by the structure of a FORTRAN source code. As a result, restructuring of the codes is necessary to assist the compiler. In the following sections, emphasis will be given to implicit vectorization over that of explicit vectorization. The advantages and disadvantages of each have already been described in Section 2.3. Various vectorization inhibitors followed by a description of techniques for vectorization are discussed in the following.

1. Avoiding calling subroutines and external functions within a loop.

Consider the loop

```
      DO 1 I=1,NR
        DO 2 J=1,NZ
          A = V(I,J)
          B = W(I,J)
          CALL ADD(A,B,C)
          F(I,J) = SQ2(C)
2       CONTINUE
1     CONTINUE
.....................................................
      SUBROUTINE ADD(A,B,C)
      C = A + B
      RETURN
      END
.....................................................
```

```
      FUNCTION SQ2(C)
      COMMON ALPHA,BETA
      SQ2 = ALPHA*SQRT(C) + BETA*SQRT(C)
      RETURN
      END
```

The above example is not vectorizable because the call to subroutine ADD and the external function SQ2 are referenced within a DO loop. If the subroutine and the external function coding is placed within the loop, this gives:

```
      DO 1 I=1,NR
       DO 2 J=1,NZ
         A = V(I,J)
         B = W(I,J)
         C = A + B
         F(I,J) = ALPHA*SQRT(C) + BETA*SQRT(C)
2      CONTINUE
1     CONTINUE
```

The above DO loop is now vectorizable because there is no reference to an external function or subroutine.

2. Increasing arithmetic chaining.

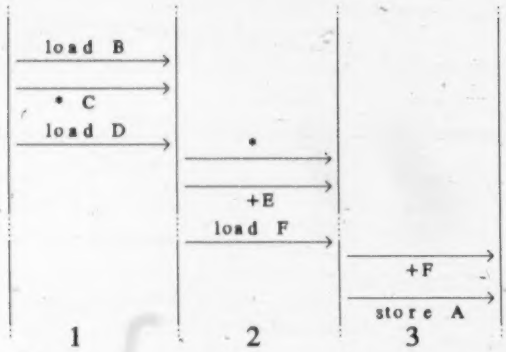
The above code will run faster in the following form because the number of arithmetic operations has been increased in the inner DO loop and unnecessary storage has been eliminated.

```
      COMMON ALPHA,BETA
      DO 1 I=1,NR
       DO 2 J=1,NZ
        C = V(I,J) + W(I,J)
        F(I,J) = ALPHA*SQRT(C) + BETA*SQRT(C)
2      CONTINUE
1     CONTINUE
```

Another way to study the chaining process is to examine the chain operation times (chimes). Consider the following example:

```
      DO 10 I=1,N
10    A(I) = B(I) * C * D(I) + E + F(I)
```

The chaining process for a CRAY can be visualized as:



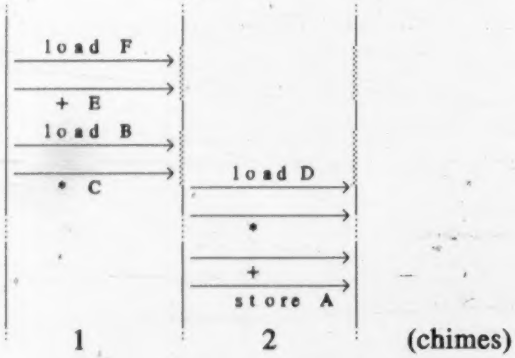
Three chimes process operates as follows:

- 1st *chime:* load B, multiplied by C and load D can occur simultaneously.
- 2nd *chime:* the result of (B*C)*D plus E can chain with the load of F.
- 3rd *chime:* The result of [(B*C)*D + E] plus F can chain with the store of result A.

This operation takes 3 chimes to complete vector result A. To increase chaining, the above example can be restructured as:

```
      DO 10 I=1,N
       A(I) = B(I) * C * D(I) + ( E + F(I) )
10    CONTINUE
```

The new chaining process can be realized as:



- 1st *chime:* load F added to E can chain with load B multiplied by C.
- 2nd *chime:* load D multiplied by (B*C) can chain with the addition of (B*C)*D and (F+E) and stored the vector result A.

In this restructuring loop, there are only two chimes to complete the operation of vector result A. In such an example, with improved vectorizability and chaining processes, the program can achieve speedup rate and performance remarkably.

3. Combining short DO loops and short subroutines.

Consider the following example containing a series of calling short subroutines:

```
      CALL VADD(A,B,C,N)
      CALL VMULT(C,A,E,N)
      CALL VADD(E,B,A,N)
.....................................................
      SUBROUTINE VADD(X,Y,Z,N)
      DO 1 I=1,N
       Z(I) = X(I) + Y(I)
1     CONTINUE
      RETURN
      END
.....................................................
      SUBROUTINE VMULT(X,Y,Z,N)
      DO 1 I=1,N
       Z(I) = X(I) * Y(I)
1     CONTINUE
      RETURN
      END
```

The example above is clearly vectorizable. However, it is important to use as fewer loops as possible, which contain long code blocks instead of many short code vectorizable blocks. The restructured code is

```
      DO 1 I=1,N
       A(I) = ( A(I) + B(I) ) * A(I) + B(I)
1     CONTINUE
```

The above restructured code is significantly faster than the

series of calls. The sum A+B and the product (A+B)*A do not have to be stored, and can be kept in a register to avoid fetching the second time. This is an example of manipulation of operations in order to increase chaining (discussed in Section 2).

Several short loops of the same length, if sharing common variables and expressions, should be combined in order to effectively utilize the machine's registers. For example:

```
      DO 1 I=1,N
       A(I) = X(I) - Y(I)*W(I) / S
1     CONTINUE
      DO 2 I=1,N
       B(I) = Y(I)*W(I) + ALPHA*X(I)
2     CONTINUE
      DO 3 I=1,N
       A(I) = A(I) + Y(I)*W(I) - B(I)
3     CONTINUE
```

The above example should be better combined as:

```
      DO 10 I=1,N
       A(I) = X(I) - Y(I)*W(I) / S
       B(I) = W(I) + ALPHA*X(I)
       A(I) = A(I) + Y(I)*W(I) - B(I)
10    CONTINUE
```

In this form, the compiler fetches the values of X,Y and W only once and then reuses them from the registers. The common sub-expression Y*W is calculated only once for each iteration and is stored in the register for further use.

4. Optimizing the innermost DO loops.

Consider an example:

```
      DO 2 I=1,5000
        DO 2 J=1,50
         A(I,J) = ( A(I,J) * B(I,J) )+ C(I,J)
2     CONTINUE
```

For the CRAY XMP, the compiler vectorizes only the innermost DO

loop; therefore, reversing the order of I and J loop would represent an improvement over the original code.

```
      DO 2 J=1,50
        DO 2 I=1,5000
         A(I,J) = ( A(I,J) * B(I,J) )+ C(I,J)
2     CONTINUE
```

5. Unrolling short DO loops.

Consider an example in which the order of I and J can not be reversed.

```
      DO 1 I=1,1000
        DO 1 J=1,4
          C(I,J) = C(I,J) +  A(I,J)*B(I,J)
1     CONTINUE
```

Some compilers can automatically unroll a short DO loop as described above. However, it is safer if the programmer unrolls the loop and helps the compiler to vectorize it. Since J index only goes from 1 to 4, It is prudent to write the above DO loops as:

```
      DO 1 I=1,1000
        C(I,1) = C(I,1) + A(I,1)*B(I,1)
        C(I,2) = C(I,2) + A(I,2)*B(I,2)
        C(I,3) = C(I,3) + A(I,3)*B(I,3)
        C(I,4) = C(I,4) + A(I,4)*B(I,4)
1     CONTINUE
```

Consider another example of unrolling a short DO loop. The original loop is:

```
      DO 20 I=1,N
      C(I) = 0.0
       DO 30 J=1,3
30       C(I) = C(I) + A(J)*B(I,J)
20    CONTINUE
```

The restructured loop is:

```
      DO 40 I=1,N
      C(I) = A(1)*B(I,1) + A(2)*B(I,2) + A(3)*B(I,3)
30    CONTINUE
```

This restructured DO loop gives the compiler more calculations to optimize, thus obtains more overlapping of functional units in the CPU, increases chaining processes and reduces memory traffic. The restructured DO loops, obviously, out-perform the original loops.

6. Eliminating unneeded storage.

Consider the following example

```
      DO 1 I=1,N
        TEMP(I) = A(I) + B(I)
        C(I)    = SQRT( TEMP(I))
        D(I)    = E(I) / TEMP(I)
1     CONTINUE
```

Suppose the variable TEMP(I) is a temporary array whose results are not needed, a scalar temporary can be used to increase performance.

```
      DO 1 I=1,N
        SCALAR  = A(I) + B(I)
        C(I)    = SQRT( SCALAR)
        D(I)    = E(I) / SCALAR
1     CONTINUE
```

The temporary variable SCALAR is promoted to a vector, while executing, the compiler treats the SCALAR as an entire vector. Only the N^{th} iteration is stored into memory. Even though it is optimal, the last result SCALAR still has to be stored into memory. A better way to handle this situation is to restructure the above DO loop as follows:

```
      DO 1 I=1,N
        D(I)    = A(I) + B(I)
        C(I)    = SQRT( D(I))
        D(I)    = E(I) / D(I)
1     CONTINUE
```

In this manner, only the vectors C(I) and D(I) are to be stored back into memory and the storage of the variable SCALAR can be

avoided. In general, the use of temporary scalar has proven to be a very useful technique in eliminating common sub-expressions and is illustrated in the following example.

7. Eliminating common sub-expressions.

Consider an example

```
      DO 1 I=1,N
        A(I) = X(I)*Y(I) + V(I)/W(I)
        B(I) = X(I)*Y(I) / Z**2
        C(I) = B(I) + V(I)/W(I)
        D(I) = X(I)*Y(I) - V(I)/W(I) / Z**2
1     CONTINUE
```

In the above DO loop, both X(I)*Y(I) and V(I)*W(I) are common sub-expressions. For each iteration, the expressions X*Y and V*W have to be computed for each calculation of A,B,C,D. The restructured loop applies the scalar temporary TEMP1 and TEMP2 in order to avoid repeated calculations.

```
      DO 1 I=1,N
        TEMP1 = X(I)*Y(I)
        TEMP2 = V(I)/W(I)
        A(I)  = TEMP1 + TEMP2
        B(I)  = TEMP1 / Z**2
        C(I)  = B(I) + TEMP2
        D(I)  = TEMP1 - TEMP2 / Z**2
1     CONTINUE
```

From the above DO loop, the compiler still has to store the variables TEMP1 and TEMP2 back into memory. This is unnecessary and may not optimally increase performance. One way to eliminate this ambiguous condition is to place parenthesis in a common expression in order to increase chaining.

8. Avoiding recurrence.

One of the most difficult problems on vector computers is the vectorization of linear and non-linear recurrences. The definition of

recursion will be made clear in the following examples. A recursive loop prevents the compiler from vectorization. Therefore, the programmer has to try to avoid or separate any loops which are recursive or even potentially recursive. Once the problem of recursiveness is addressed and, if no implicit vectorization is possible, special calls to functions and subroutines must be used to resolve the recurrence. The CFT FORTRAN compiler and FT-200 FORTRAN compiler [45,46] provide many routines and special functions to vectorize recursive DO loops. In this thesis, explicit vectorization is not considered.

When using implicit vectorization, one way to address the recursive problems is to separate the recursive and non-recursive parts. Consider the following loop:

```
      D(1) = 1.0
      E(1) = 2.0
      DO 10 I=2,N
       D(I) = B(I) * A(I) + TEMP1
       E(I) = C(I) * B(I) - TEMP2
       F(I) = ( D(I-1) * A(I-1) ) / TEMP1
       G(I) = ( E(I-1) * B(I-1) ) / TEMP2
10     CONTINUE
```

The above DO loop is recursive since the vectors F and G depend on D and E, respectively. The result for D and E must be determined first before F and G can be calculated. In other words, results for the vectors F(I) and G(I) depend on value D(I-1) and E(I-1), respectively. In order to increase performance, vector D and E can be computed first in a separate DO loop before any calculation of F and G taken place. By separating vectors D,E from F,G which are coupled to one another, one can take advantage of supervector speed instead of

preventing the whole loop from being vectorizable by recursion. A possible restructuring for vectorization is given as:

```
      DO 10 I=2,N
       D(I) = B(I) * A(I) + TEMP1
       E(I) = C(I) * B(I) - TEMP2
10    CONTINUE
      D(1) = 1.0
      E(1) = 2.0
      DO 20 I=2,N
       F(I) = ( D(I-1) * A(I-1) ) / TEMP1
       G(I) = ( E(I-1) * B(I-1) ) / TEMP2
20    CONTINUE
```

9. Restructuring IF statements.

Most loops containing IF statements may prevent the compiler from vectorization. Vectorizing these IF statements can be cumbersome; therefore, great care should be taken depending on the structure of the codes. Consider an example:

```
      D0 1 K=2,N
        DO 2 J=2,N
          DO 3 I=2,N
            A(I,J) = PX * ( B(I+1,J,K) + B(I-1,J,K )
                   + PY * ( B(I,J+1,K) + B(I,J-1,K )
                   + PZ * (B(I,J,K+1) + B(I,J,K-1 )
          IF (K .LT. 5) THEN
              C(I,J) = A(I,J)
          ELSE
              C(I,J) = A(I,J)+ 1.0
          ENDIF
3         CONTINUE
2       CONTINUE
1     CONTINUE
```

In the above DO loops, the elements of array A are computed conditionally. The condition depends only on value of the outer DO loop K. These DO loops are, however, still vectorized, but the conditional statement within the innermost DO loop may reduce the performance level. Also, the condition for optimal vectorization is to make the innermost DO loop most efficient and to avoid conditional

statements as much as possible. Therefore, the loop can be separated making the innermost DO (I index)loop more efficiently by forcing the IF statement to the outer DO loop (K index). The restructuring will be:

```
      DO 1 K=2,N
        DO 2 J=2,N
          DO 2 I=2,N
            A(I,J) = PX * ( B(I+1,J,K) + B(I-1,J,K )
                   + PY * ( B(I,J+1,K) + B(I,J-1,K )
                   + PZ * (B(I,J,K+1) + B(I,J,K-1 )
2         CONTINUE
        IF (K .LT. 5) THEN
          DO 3 J=2,N
            DO 3 I=2,N
              C(I,J) = A(I,J)
3         CONTINUE
        ELSE
          DO 4 J=2,N
            DO 4 I=2,N
              C(I,J) = A(I,J) + 1.0
4         CONTINUE
        ENDIF
1     CONTINUE
```

10. Avoiding I/O statements.

The appearance of I/O statements within a DO loop can be treated as the reference to external functions or subroutines. Therefore, I/O statements will prevent optimization for the rest of the code within a loop. The programmer should use I/O statements only when necessary. If I/O statements must be used within a loop, they should be separated into another loop.

Consider the original loop:

```
      DO 10 I=1,N
        DO 20 J=1,M
          C(I,J) = A(I,J) + B(I,J)
          D(I,J) = ( A(I,J) * B(I,J) ) / SCALAR
          WRITE(6,30) C(I,J),D(I,J)
20        CONTINUE
10      CONTINUE
```

The above DO loops are not vectorizable because of the I/O WRITE statement. In order to vectorize the above DO loop, it can be split into a vectorizable and a non-vectorizable part as:

```
      DO 10 I=1,N
        DO 20 J=1,M
          C(I) = A(I,J) + B(I,J)
          D(I) = A(I,J) * B(I,J)
20      CONTINUE
10    CONTINUE
      DO 1 I=1,N
        DO 2 J=1,M
          WRITE(6,30) C(I,J),D(I,J)
2       CONTINUE
1     CONTINUE
```

11. Avoiding the use of assigned GO TO statements.

The use of GO TO statements contain hidden information that cannot be known at the compilation time. Therefore, a code which has GO TO statements cannot be optimized. There are several ways to eliminate the use of GO TO statements. One approach is to use the conditional IF and ELSE statements. Another approach is to use the DO loop structure. Consider an example that contains the GO TO statements.

```
      DO 1 I = 1,N
      GO TO (2,3) I
 2    D(I) = B(I)
      A(I) = D(I) + ALPHA
      GO TO 1
 3    D(I) = C(I)
      A(I) = D(I) + BETA
 1    CONTINUE
```

To eliminate GO TO statements in the above example, the conditional IF and ELSE is used to restructure the code as:

```
      DO 1 I = 1,N
        IF (I .NE. 2) THEN
          D(I) = B(I)
          A(I) = D(I) + ALPHA
        ELSE
```

```
            D(I) = C(I)
            A(I) = D(I) + BETA
          ENDIF
1       CONTINUE
```

The above loop is clearly vectorizable for the CRAY-XMP (not necessary for other types such as the CYBER 205), since the condition is nested within the loop. The following illustration will utilize the use of a DO loop instead of GO TO statements.

```
          I = 0
1         CONTINUE
          I = I + 1
          A(I) = ( B(I)**2 + C(I)*D(I) ) / E(I)
          IF (I .LT. N) GO TO 1
```

The above example contains a backward GO TO statement. It prevents the compiler from optimizing this code. The following restructured code will be clearly vectorizable.

```
          DO 1 I = 1,N
           A(I) = ( B(I)**2 + C(I)*D(I) ) / E(I)
1         CONTINUE.
```

2.7 VECTORIZATION ON THE CYBER 205

The CYBER 205 is a memory-to-memory vector computer whose architecture has a different implication for implementing vectorization than those used for the CRAY XMP computer.

The first point to note is that the CYBER 205 has a fixed start-up time that is required to set the segmented general purpose pipelines for the vectors. The summary of start-up time for all operations is outlined in Table 1. The start-up time for the CYBER 205 is independent of the vector length, thus processing of short vectors is completely dominated by the start up time for each instruction. The performance will increase as the function of the vector length

increases. One disadvantage of the CYBER 205 is that the vector length should not exceed 2^{16} elements for the total nested loops in order to take advantage of vectorization.

Secondly, it is noted that the definition of a vector or array on the CYBER 205 is a contiguous set of memory locations. The need for long and contiguous stored vector influences the process of vectorization greatly. Therefore, it is no longer sufficient to only vectorize the innermost DO loops. As a result, storing the multi-dimensional arrays into a long and contiguous vector is more efficient than vectorizing only the innermost DO loop of a multi-dimensional array. In conclusion, the main difficulty for vectorization on the CYBER 205 is the construction of long and contiguously stored vectors.

The basic techniques, methods and vectorizing inhibitors for the CYBER 205 are similar to that for the CRAY XMP discussed in the previous sections. There are some special features which distinguish it from the CRAY XMP computers. These include:

1. Link triad.

The most common expressions occurring in programming usually are the ADD and MULTIPLY operations. Under many circumstances, the add and multiply pipelines can be linked together, from which one produces a result and feeds directly into other simultaneously, thus again doubling the result rate. Consider an example:

```
      DO 1 I=1,N
        A(I) = B(I)*C(I) + D(I)
1     CONTINUE
```

The link triad feature from the above DO loop can be interpreted

as follows. The vector result from B*C can be linked with vector D in the adder and the resultant vector A can flow to memory simultaneously. This operation is similar to the chaining process occurring in the CRAY XMP, from which the intermediate result of vector B*C is retained in the registers until the vector D has arrived for addition. However, the chaining that occurs in the CRAY XMP is more flexible and complicated.

2. Gather/Scatter instructions.

The vector pipeline instructions of the CYBER 205 always operate on contiguous data. Therefore, the data reference with the stride (increment or decrement in indices) other than one need some special treatment before entering the pipeline. In other words, the data is gathered and then converted into a contiguous stream. Once the results are computed, they need to be scattered back into their original form. This operation is called gather and scatter instructions of the data stream.

Consider an example of a DO loop with a stride of 5:

```
      DO 1 I=1,N,5
        A(I) = B(I) + C(I)
1     CONTINUE
```

The vector pipeline cannot directly fetch or store every fifth item of data. The vector stream unit issues a gather instruction to fetch and store the data of B and C in a temporary vector in a contiguous way. Once the result of B+C is computed, the scatter instruction is issued to store every fifth element into array A. The following illustration interprets the gather/scatter instruction as follows:

```
C Gather every fifth element of B and C
      J = 0
      DO 1 I=1,N,5
       J = J + 1
       TEMPB(J) = B(I)
       TEMPC(J) = C(I)
1     CONTINUE
C Performing the addition of the gathered arrays
      DO 2 K=1,J
       TEMPA(K) = TEMPB(K) + TEMPC(K)
2     CONTINUE
C Scatter the result back into A
      DO 3 I=1,N,5
       J = J + 1
       A(I) = TEMPA(J)
3     CONTINUE.
```

The loops containing stride other than one are vectorizable for the CYBER 205 since its compiler can instruct the gathering and scattering processes. However, the CRAY XMP's compiler cannot vectorize loops that have stride other than one.

3. Problems on multi-dimensional arrays.

Consider an example:

```
      DO 1 J=1,N
      D0 1 I=1,N
        A(I,J) = B(I,J) + C(I,J)
1     CONTINUE
```

The above DO loops can vectorize not only the inner DO loop but also the outer DO loop because the total arrays are stored contiguously in the memory and fetched contiguously into the pipeline.

Consider another example:

```
      DO 1 J=2,N-1
      D0 1 I=2,N-1
        A(I,J) = B(I,J) + C(I,J)
1     CONTINUE
```

Clearly, the compiler can vectorize the inner DO loop with vector length of N-2 only. Even though the total data elements are stored

contiguously in the memory, the active elements of each array are not. Further vectorization in this case is not straightforward and it can only be overcome through explicit declaration and by calling special subroutines. The detail of explicit vectorization can be found in [46]. The aim of the present research is focused mainly on the implicit vectorization.

2.8 VECTORIZATION SUMMARY

The vectorization procedures for both the CRAY XMP and CYBER 205 computers can be summarized as follows:

Make:

- *the innermost* DO *loops most efficient.*

Avoid:

- IF *statements as much as possible.*
- *assigned* GO TO *statements.*
- *unnecessary* I/O *statements.*
- *calling subroutines and functions within a loop.*
- *irregular (or non-linear) addressing.*
- *any memory bank conflicts.*
- *complicate branching within a* DO *loop.*
- *linear and non-linear recursions.*
- *unnecessary storage and common sub-expression by introducing scalar temporary.*

Use:

- *many arithmetic operations within inner loops instead of many loops with few operations.*

- *long vectors.*
- *chaining possibilities and link triads to achieve supervector speed.*

Unroll:

- *short* DO *loops.*

Separate:

- *vectorizable parts from non-vectorizable parts.*

Switch:

- *inner and outer loops to have a longer index range on the inner loop and to suppress dependencies.*

Some important points concerning vectorization on the CRAY XMP and CYBER 205 computers are:

- *For the* CYBER 205, *change* 2D *and* 3D *arrays to* 1D *arrays in order to have long contiguously stored vectors.*
- *Avoid gather/scatter operation on a* CRAY *computers.*
- *Employ chaining to use registers effectively on the* CRAY XMP.
- *Use link triads on the* CYBER 205.

CHAPTER THREE
LASER PROPAGATION MODEL: THEORY AND NUMERICAL SOLUTION

The HEATER code, originally developed by J. N. McMullin, C. E. Capjack and C. R. James [48], describes the propagation of a laser beam in a cylindrically symmetric plasma by numerically solving the Helmholtz wave equation. The code accounts for the effects of refraction, diffraction, beam incoherency and absorption by a plasma medium. HEATER has been designed to be an independent laser propagation package and can be ported to any Eulerian and Lagrangian hydrodynamic code. The input values read by HEATER include the electron temperature and density, ion temperature and density, input power, and laser parameters (pulse shape, wavelength). All calculations are spatially dependent on the laser flux, the absorbed power, the radial and axial ponderomotive forces. These forces are important in the study of laser self-focusing and filamentation.

3.1 THEORY AND EQUATIONS

3.1.1 The wave equation.

The time independent wave equation for electromagnetic wave propagation through a plasma medium is described as follows [51,52]:

$$\nabla^2 \mathbf{E} + \frac{\omega^2}{c^2} \varepsilon \mathbf{E} = 0 \qquad (3.1)$$

In the unmagnetized (unperturbed) plasma, if the term $\nabla.\mathbf{E}$ can be neglected (App. 1), the above equation is written as:

$$\nabla^2 E + \left(\frac{\omega}{c}\right)^2 \varepsilon E = 0 \qquad (3.2)$$

In the limit where $\omega >> \omega_p$, the dielectric constant ε is approximately given by:

$$\varepsilon = \varepsilon(r,z) = 1 - \frac{\omega_p^2(r,z)}{\omega^2}\left[1 - i\frac{\upsilon_c}{\omega}\right] \qquad (3.3)$$

where ω = laser frequency

ω_p = plasma frequency

υ_c = collision frequency.

3.1.2 The WKB approximation to a wave equation.

In cylindrical coordinates, equation (3.2) can be expressed as:

$$\frac{\partial^2 E}{\partial^2 z} + \nabla_\perp^2 E + \left(\frac{\omega}{c}\right)^2 \varepsilon E = 0 \qquad (3.4)$$

where $\nabla_\perp^2$ is the transverse Laplacian operator and is given by:

$\nabla_\perp^2 = \frac{1}{r}\frac{\partial}{\partial r}\left[r\frac{\partial}{\partial r}\right]$ when axial symmetry is assumed.

For a laser beam travelling in the positive z direction, the fast and slow phase variation in the electric field can be separated as follows:

$$E(r,z) = \left[\frac{8\pi\omega}{k(z)c^2}\right]^{1/2} F(r,z)\, e^{i\int_0^z k(z')dz'} \qquad (3.5)$$

where k(z) is defined by

$$k(z) = \frac{\omega}{c}\left[\mathrm{Re}\{\varepsilon(o,z)\}\right]^{1/2} = \frac{\omega}{c}\left[1 - \frac{\omega_p^2(o,z)}{\omega^2}\right]^{1/2} \qquad (3.6)$$

Equation (3.5) contains both the slow, [F(r,z)], and fast, $e^{i\int_0^z k(z')dz'}$, phase variations.

$$\text{Let } E(r,z) = \frac{A}{\sqrt{k(z)}} F(r,z)\, e^{i\int_0^z k(z')dz'} \tag{3.7}$$

where $A = \sqrt{\frac{8\pi\omega}{c^2}}$ is a constant and

$$F = F(r,z) \tag{3.8}$$

By using expression (3.7) for E_z, the term $\frac{\partial^2 E}{\partial z^2}$ becomes:

$$\frac{\partial^2 E}{\partial z^2} = \underbrace{\frac{\partial}{\partial z}\left[AF\frac{\partial}{\partial z}\left(\frac{1}{\sqrt{k(z)}}\right)e^{i\int_0^z k(z')dz'}\right]}_{(A)} + \underbrace{\frac{\partial}{\partial z}\left[\frac{A}{\sqrt{k(z)}}\frac{\partial F}{\partial z}e^{i\int_0^z k(z')dz'}\right]}_{(B)}$$

$$+ \underbrace{\frac{\partial}{\partial z}\left[iAF\sqrt{k(z)}e^{i\int_0^z k(z')dz'}\right]}_{(C)} \tag{3.9}$$

The evaluation of the partial derivative of term **A** with respect to z gives:

$$\frac{\partial}{\partial z}\left[AF\frac{\partial}{\partial z}\left(\frac{1}{\sqrt{k(z)}}\right)e^{i\int_0^z k(z')dz'}\right] = Ae^{i\int_0^z k(z')dz'}\left[\frac{\partial F}{\partial z}\frac{\partial}{\partial z}\left(\frac{1}{\sqrt{k(z)}}\right) + F\frac{\partial^2}{\partial z^2}\left(\frac{1}{\sqrt{k(z)}}\right) + \frac{\partial}{\partial z}\left(\frac{1}{\sqrt{k(z)}}\right)ik(z)F\right] \tag{3.10}$$

An evaluation of $\frac{\partial}{\partial z}$ [term **B**] in Eq. (3.9) gives:

$$\frac{\partial}{\partial z}\left[\frac{A}{\sqrt{k(z)}}\frac{\partial F}{\partial z}e^{i\int_0^z k(z')dz'}\right] = Ae^{i\int_0^z k(z')dz'}\left[\frac{\partial F}{\partial z}\frac{\partial}{\partial z}\left(\frac{1}{\sqrt{k(z)}}\right) + \frac{1}{\sqrt{k(z)}}\frac{\partial^2 F}{\partial z^2} + i\sqrt{k(z)}\frac{\partial F}{\partial z}\right] \tag{3.11}$$

An evaluation of $\frac{\partial}{\partial z}$ [term **C**] in Eq. (3.9) yields:

$$\frac{\partial}{\partial z}\left[iAF\sqrt{k(z)}e^{i\int_0^z k(z')dz'}\right] = iAe^{i\int_0^z k(z')dz'}\left[\sqrt{k(z)}\frac{\partial F}{\partial z} + F\frac{\partial}{\partial z}(\sqrt{k(z)})\right.$$

$$\left. + iFk(z)^{3/2}\right] \qquad (3.12)$$

If Eqs. (3.10), (3.11), (3.12) are combined, then

$$\frac{\partial^2 E}{\partial z^2} = Ae^{i\int_0^z k(z')dz'}\left\{\left[\frac{\partial F}{\partial z}\frac{\partial}{\partial z}\left(\frac{1}{\sqrt{k(z)}}\right) + F\frac{\partial^2}{\partial z^2}\left(\frac{1}{\sqrt{k(z)}}\right) + \frac{\partial}{\partial z}\left(\frac{1}{\sqrt{k(z)}}\right) ik(z)\right]\right.$$

$$+ \left[\frac{\partial F}{\partial z}\frac{\partial}{\partial z}\left(\frac{1}{\sqrt{k(z)}}\right) + \frac{\partial^2 F}{\partial z^2}\left(\frac{1}{\sqrt{k(z)}}\right) + \frac{\partial F}{\partial z}(i\sqrt{k(z)})\right]$$

$$\left. + i\left[\sqrt{k(z)}\frac{\partial F}{\partial z} + F\frac{\partial}{\partial z}(\sqrt{k(z)}) + iFk(z)^{3/2}\right]\right\} \qquad (3.13)$$

$$= Ae^{i\int_0^z k(z')dz'}\left\{2\frac{\partial F}{\partial z}\frac{\partial}{\partial z}\left(\frac{1}{\sqrt{k(z)}}\right) + 2i\sqrt{k(z)}\frac{\partial F}{\partial z} + F\frac{\partial^2}{\partial z^2}\left(\frac{1}{\sqrt{k(z)}}\right)\right.$$

$$\left. + iF\sqrt{k(z)}\frac{\partial}{\partial z}\left(\frac{1}{\sqrt{k(z)}}\right) + \frac{\partial^2 F}{\partial z^2}\left(\frac{1}{\sqrt{k(z)}}\right) + iF\frac{\partial}{\partial z}(\sqrt{k(z)}) - Fk(z)^{3/2}\right\} \qquad (3.14)$$

Substituting Eq. (3.14) into Eq. (3.4) and multiplying both sides by $1/(Ae^{i\int_0^z k(z')dz'})$, it yields:

$$F\frac{\partial^2}{\partial z^2}\left(\frac{1}{\sqrt{k(z)}}\right) + iFk(z)\frac{\partial}{\partial z}\left(\frac{1}{\sqrt{k(z)}}\right) + 2\frac{\partial F}{\partial z}\frac{\partial}{\partial z}\left(\frac{1}{\sqrt{k(z)}}\right) + \frac{1}{\sqrt{k(z)}}\frac{\partial^2 F}{\partial z^2}$$

$$+ i2\sqrt{k(z)}\frac{\partial F}{\partial z} + iF\frac{\partial}{\partial z}(\sqrt{k(z)}) - Fk(z)^{3/2} + \frac{1}{\sqrt{k(z)}}\nabla_\perp{}^2 F$$

$$+ \left(\frac{\omega}{c}\right)^2\sqrt{k(z)}\,\varepsilon\, F = 0 \qquad (3.15)$$

The WKB approximation is found to be valid when the following

inequalities are satisfied:

$$\frac{\partial^2}{\partial z^2}\left[\frac{1}{\sqrt{k(z)}}\right] \ll \frac{\partial}{\partial z}\left[\frac{1}{\sqrt{k(z)}}\right]$$

$$\frac{\partial}{\partial z}\left[\frac{1}{\sqrt{k(z)}}\right] \ll \frac{1}{\sqrt{k(z)}}$$

$$\frac{\partial}{\partial z}(\sqrt{k(z)}) \ll \sqrt{k(z)}$$

The WKB approximation [51,52] assumes that variations in F, ω_p and υ_c over one wave length are small. In other words, the WKB condition restricts the refractive index to a relatively small spatial variation. The second-order terms are usually not taken into account and terms such as $\frac{\partial^2 F}{\partial z^2}$, $\frac{\partial^2}{\partial z^2}\left[\frac{1}{\sqrt{k(z)}}\right]$, $\frac{\partial}{\partial z}\left[\frac{1}{\sqrt{k(z)}}\right]$ and $\frac{\partial}{\partial z}(\sqrt{k(z)})$ are neglected. The expression (3.15) reduces to:

$$- Fk(z)^{3/2} + i2\sqrt{k(z)}\frac{\partial F}{\partial z} + \frac{1}{\sqrt{k(z)}}\nabla\perp^2 F + \left(\frac{\omega}{c}\right)^2 \varepsilon \frac{F}{\sqrt{k(z)}} = 0 \quad (3.16)$$

$$\frac{\partial F}{\partial z} = \frac{i}{2k(z)}\left[\nabla\perp^2 F - q^2(r,z)\, F\right] \quad (3.17)$$

where

$$q^2(r,z) = k(z)^2 - \frac{\omega^2}{c^2}\varepsilon \quad (3.18)$$

The component of the complex field amplitude, F(r,z), is related to the time-averaged electromagnetic flux $\mathcal{F}$ [48] by the relation:

$$\mathcal{F}(r,z) = |F(r,z)|^2 \quad (3.19)$$

By substituting Eqs (3.3) and (3.6) into Eq. (3.18), the term $q^2(r,z)$ can be expressed as:

$$q^2(r,z) = -\frac{\omega^2}{c^2}\left[1 - \frac{\omega_p^2(r,z)}{\omega^2}\left[1 - i\frac{\upsilon_c}{\omega}\right]\right] + \frac{\omega^2}{c^2}\left[1 - \frac{\omega_p^2(0,z)}{\omega^2}\right]$$

$$= -\frac{1}{c^2}\left[\ \omega^2 - \omega_p^2(r,z) - i\omega_p^2(r,z)\frac{\upsilon_c}{\omega} - \omega^2 + \omega_p^2(0,z)\ \right]$$

$$= \frac{1}{c^2}\left[\ \omega_p^2(r,z) - \omega_p^2(0,z) - i\omega_p^2(r,z)\frac{\upsilon_c}{\omega}\ \right] \qquad (3.20)$$

For convenience, the spatial absorption coefficient $K_{abs}(r,z)$ [53] is introduced. and the term $q^2(r,z)$ [48] is written in the following form:

$$q^2(r,z) = \frac{1}{c^2}\left[\ \omega_p^2(r,z) - \omega_p^2(0,z)\ \right] - ik(z)K_{abs}(r,z) \qquad (3.21)$$

$$\text{where } K_{abs}(r,z) = \frac{13.51\ Z\ \text{LogC}\left(\frac{N_e}{N_c}\right)^2}{\lambda^2\left[\ 1 - \frac{N_e}{N_c}\ \right]^{1/2} T_e^{3/2}} \qquad (3.22)$$

N_e is the electron density

N_c is the critical density and $N_c = \frac{1.1 \times 10^{15}}{\lambda^2}$.

LogC is the Coulomb logarithm given by Dawson and Johnston [53] where $C = \min\left(\frac{v_t}{\omega_p p_{min}}, \frac{v_t}{\omega p_{min}}\right)$, v_t is the thermal velocity and p_{min} is the minimum impact parameter for the electron-ion collision. The ratio $\frac{N_e}{N_c}$ is related to the plasma and laser frequency by

$$\frac{N_e}{N_c} = \frac{\omega_p^2}{\omega^2} \qquad (3.23)$$

From Eq. (3.23), Eq. (3.21) can also be written in terms of the ratio N_e/N_c as:

$$q^2(r,z) = \frac{\omega^2}{c^2}\left[\ \frac{N_e(r,z)}{N_c} - \frac{N_e(0,z)}{N_c}\ \right] - ik(z)K_{abs}(r,z) \qquad (3.24)$$

3.1.3 The ponderomotive force equations.

The ponderomotive force is given by [54]: (Appendix 2)

$$f = - \frac{\omega_p^2 \nabla \langle E^2 \rangle}{\omega^2 \, 8\pi} \tag{3.25}$$

If the fields are assumed symmetrically (ie. ϕ independent), then the ponderomotive forces in the radial and axial direction become:

$$f_r = - \frac{\omega_p^2}{\omega^2} \frac{\partial}{\partial r} \frac{\langle E^2 \rangle}{8\pi} \tag{3.26}$$

$$f_z = - \frac{\omega_p^2}{\omega^2} \frac{\partial}{\partial z} \frac{\langle E^2 \rangle}{8\pi} \tag{3.27}$$

Since $\frac{1}{2}\langle E^2 \rangle = EE^*$, from Eq. (3.5), $\langle E^2 \rangle$ can be expressed as:

$$E(r,z)E(r,z)^* = \frac{8\pi\omega}{k(z)\,c^2} |F(r,z)|^2 \tag{3.28}$$

The substitution of Eq. (3.28) into Eqs. (3.26) and (3.27) yields:

$$f_r = - \frac{\omega_p^2}{2\omega} \frac{\partial}{\partial r} \frac{|F(r,z)|^2}{k(z)c^2} \tag{3.29}$$

$$f_z = - \frac{\omega_p^2}{2\omega} \frac{\partial}{\partial z} \frac{|F(r,z)|^2}{k(z)c^2} \tag{3.30}$$

If the group velocity is defined by $v_g(z) = \frac{k(z)c^2}{\omega}$, Eqs. (3.29) and (3.20) can be expressed in terms of group velocity as:

$$f_r = - \frac{\omega_p^2}{2\omega^2} \frac{\partial}{\partial r} \left[\frac{|F(r,z)|^2}{v_g(z)} \right] \tag{3.31}$$

$$f_z = - \frac{\omega_p^2}{2\omega^2} \frac{\partial}{\partial z} \left[\frac{|F(r,z)|^2}{v_g(z)} \right] \tag{3.32}$$

From Eq. (3.23), the above equations can be expressed in terms of electron density ratio as:

$$f_r(r,z) = - \frac{n(r,z)}{2n_c v_g(z)} \frac{\partial}{\partial r} |F(r,z)|^2 \tag{3.33}$$

$$f_z(r,z) = -\frac{n(r,z)}{2n_c}\left[\frac{-\frac{\partial}{\partial z}\left[v_g(z)\right]|F(r,z)|^2 + \frac{\partial}{\partial z}\left[|F(r,z)|^2\right]v_g(z)}{v_g(z)^2}\right]$$

$$-\frac{n(r,z)}{2n_c v_g(z)^2}|F(r,z)|^2\frac{\partial}{\partial z}\left[v_g(z)\right] - \frac{n(r,z)}{2n_c v_g(z)}\frac{\partial}{\partial z}|F(r,z)|^2 \qquad (3.34)$$

In calculating ponderomotive forces, the group velocity on axis is used (ie. expression involve the term k(z)). The on axis group velocity expressed in terms of ratio n/n_c as

$$v_g(z) = \frac{k(z)c^2}{\omega} = c\left[1 - \frac{\omega_p^2(0,z)}{\omega^2}\right]^{1/2} = c\left[1 - \frac{n(0,z)}{n_c}\right]^{1/2} \qquad (3.35)$$

The evaluation of the term $\frac{\partial}{\partial z}\left[v_g(z)\right]$ in Eq. (3.35) yields

$$\frac{\partial}{\partial z}\left[v_g(z)\right] = -\frac{c}{2n_c}\left[1 - \frac{n(0,z)}{n_c}\right]^{-1/2}\frac{\partial n(0,z)}{\partial z} \qquad (3.36)$$

The first term in Eq. (3.34) can be expressed by substituting Eq. (3.36) into (3.34) as follows:

$$\frac{n(r,z)}{2n_c v_g(z)^2}|F(r,z)|^2\frac{\partial}{\partial z}\left[v_g(z)\right]$$

$$= -\frac{n(r,z)}{2n_c v_g(z)^2}|F(r,z)|^2\frac{c}{2n_c}\left[1 - \frac{n(0,z)}{n_c}\right]^{-1/2}\frac{\partial n(0,z)}{\partial z}$$

$$= -\frac{1}{4}\left[\frac{n(r,z)}{n_c v_g(z)}\right]\left[\frac{c^2}{v_g(z)^2 n_c}\right]|F(r,z)|^2\frac{\partial n(0,z)}{\partial z} \qquad (3.37)$$

The axial ponderomotive force is written in term of density ratio n/n_c as:

$$f_z(r,z) = -\frac{1}{2}\frac{n(r,z)}{n_c v_g(z)}\frac{\partial}{\partial z}|F(r,z)|^2$$

$$-\frac{1}{4}\left[\frac{n(r,z)}{n_c v_g(z)}\right]\left[\frac{c^2}{v_g(z)^2 n_c}\right]|F(r,z)|^2\,\frac{\partial n(0,z)}{\partial z} \qquad (3.38)$$

Equations (3.33) and (3.38) represent radial and axial ponderomotive forces, respectively. These forces are expressed explicitly in terms of the ratio of the electron density to the critical density because this form is convenient to implement. The assumption used throughout HEATER is that the density ratio n/n_c does not exceed 0.99. Base upon this assumption, the group velocity v_g should always be greater than 0.1 c (speed of light).

3.1.4 The absorption equation.

In Eq. (3.17), the first term on the right hand side refers to diffraction of the beam and the second term to refraction and absorption. From a classical theory, the plasma absorbs laser energy by way of the inverse bremsstrahlung process, where electrons in the electric field of laser light collide with plasma electrons and ions. As a result of energy transfer due to collisions, plasma temperature increases. The inverse bremsstrahlung absorption coefficient is given by Johnson and Dawson [53]. The power absorption is equal to the incident power multiplied by the absorption coefficient (Eq. 3.22). The incident power at any point in the plasma column per unit area per unit time is given by Eq. 3.19. Therefore, the power absorption per unit volume per unit time is expressed as:

$$W_{abs}(r,z) = K_{abs}(r,z)\,|F(r,z)|^2 \qquad (3.39)$$

3.2 NUMERICAL SOLUTION

In order to simulate the laser beam propagation through the plasma media, it is necessary to solve the paraxial equation (3.17) which is

$$\frac{\partial F}{\partial z} = \frac{i}{2k(z)}\left[\frac{1\partial}{r\partial r}\left(r\,\frac{\partial F}{\partial r}\right) - q^2(r,z)F\right] \qquad (3.40)$$

From Eq. (3.40), the complex flux F is determined numerically by using cubic spline approximation in the radial direction and finite differencing in z direction. The differencing operation results in a tridiagonal matrix. Once the complex flux F is found at a given point in r and z, the real time-averaged flux $\mathbb{F}$ can be computed by using Eq. (3.19). The radial (f_r) and axial (f_z) ponderomotive forces are calculated by using relations (3.34) and (3.38), respectively. The absorption of laser energy by the plasma is computed from expression (3.39).

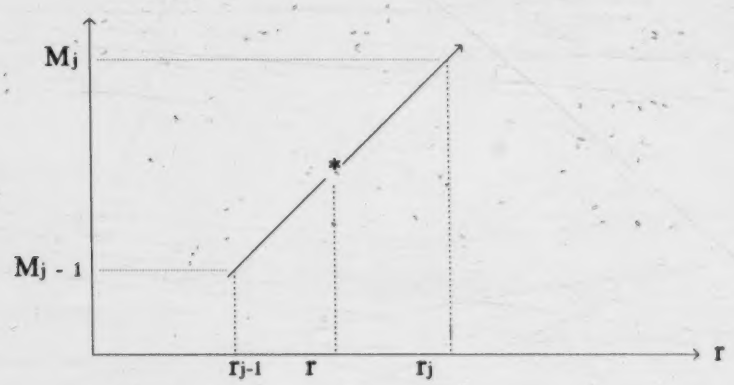
3.2.1 Radial discretization.

Before advancing the solution for the complex flux F in the z direction, it is necessary to know the radial profile of F at the previous z point. Fleck [49] has used a cubic spline method to describe the complex flux density of the beam in the radial direction. This method has proven to be accurate and flexible because it can take the use of variable mesh spacing in the radial direction. This in turn enables the boundary effects to be reduced to a minimum. An advantage of this method is that a larger mesh can be used at the outer boundary where the beam intensity is low and a smaller mesh can be used to resolve the beam near the axis where the intensity is high. An

assumption of this method is that the radial Laplacian is both piece-wise linear and continuous. Based upon this assumption, the Laplacian is given as

$$\frac{1}{r}\frac{\partial}{\partial r}\left[r\frac{\partial F}{\partial r}\right] = \frac{M_{j-1}}{l_j}(r_j - r) + \frac{M_j}{l_j}(r - r_{j-1}) \qquad (3.41)$$

for $r_{j-1} < r < r_j$ and $j = 1,2,3,\ldots,j_m$ and $l_j = r_j - r_{j-1}$.



The linear relationship between F and the Laplacian M $\left\{\frac{1}{r}\frac{\partial}{\partial r}\left[r\frac{\partial F}{\partial r}\right]\right\}$ is shown [49]:

$$A_jM_{j+1} + B_jM_j + C_jM_{j-1} = -\frac{\gamma_{j+1}}{r_j}(F_{j+1} - F_j) + \frac{\gamma_j}{r_j}(F_j - F_{j-1}) \qquad (3.42)$$

or

$$C_jM_{j-1} + B_jM_j + A_jM_{j+1} = -\frac{\gamma_j}{r_j}F_{j-1} + \left[\frac{\gamma_{j+1}}{r_j} + \frac{\gamma_j}{r_j}\right]F_j - \frac{\gamma_{j+1}}{r_j}F_{j+1} \qquad (3.43)$$

where A_j, B_j, C_j and γ_j are defined in Appendix 4 and $j = 1,2,3,\ldots,j_m$. Clearly, A, B and C can be identified as coefficients of a tridiagonal matrix M on the left hand side while $-\frac{\gamma_j}{r_j}$, $\left[\frac{\gamma_{j+1}}{r_j} + \frac{\gamma_j}{r_j}\right]$, and $-\frac{\gamma_{j+1}}{r_j}$ are coefficients of tridiagonal matrix F on the right hand side. Eq. (3.43) can be written in vector form as:

$$T_1M = T_2F \qquad (3.44)$$

where T_1 and T_2 are tridiagonal matrices.

3.2.2 Beam advection in axial direction.

Once a cubic spline has been constructed in the radial direction, the Crank-Nicolson [35] technique is used to solve the beam advection in the axial direction. The use of Crank-Nicolson difference scheme has proven to exhibit both accuracy and stability. A description of the method can be found in most numerical text books [35].

Denoting the term $\frac{1}{r}\frac{\partial}{\partial r}\left[r\frac{\partial F}{\partial r}\right]$ by M at any point r_j, the finite difference approximation at any given grid point n in the z direction is

$$\frac{\partial F}{\partial z} = \frac{F^{n+1} - F^n}{\Delta z} \tag{3.45}$$

$$\frac{1}{r}\frac{\partial}{\partial r}\left[r\frac{\partial F}{\partial r}\right] = \frac{1}{2}\left[M^n + M^{n+1}\right] \tag{3.46}$$

$$QF = \frac{1}{2}\left[F^nQ^n + F^{n+1}Q^{n+1}\right] \tag{3.47}$$

where $Q = q^2(r,z)$ and $\Delta z = z_{n+1} - z_n$.

From Eqs. (3.45), (3.46) and (3.47), the differencing of the paraxial equation (3.40) in the z direction can be written at any z-grid point n as:

$$\frac{F^{n+1} - F^n}{\Delta z} = \frac{i}{2k}\left[\frac{1}{2}\left(M^n + M^{n+1}\right) - \frac{1}{2}\left(Q^{n+1}F^{n+1} + Q^nF^n\right)\right] \tag{3.48}$$

or

$$\frac{4k}{i\Delta z}\left(F^{n+1} - F^n\right) = \left(M^n + M^{n+1}\right) - \left(Q^{n+1}F^{n+1} + Q^nF^n\right) \tag{3.49}$$

By defining $K = \frac{4k}{i\Delta z}$, the above expression is expressed as:

$$F^{n+1}(K + Q^{n+1}) - F^{n}(K - Q^{n}) = M^{n} + M^{n+1} \qquad (3.50)$$

By using relation (3.44) to eliminate M, the above expression becomes:

$$T_1F^{n+1}(K + Q^{n+1}) - T_1F^{n}(K - Q^{n}) = T_2F^{n} + T_2F^{n+1} \qquad (3.51)$$

or

$$\left[T_2 - T_1(K + Q^{n+1})\right]F^{n+1} = \left[-T_2 - T_1(K - Q^{n})\right]F^{n} \qquad (3.52)$$

This can be written in a vector form as follows:

$$T_3\, F^{n+1} = T_4\, F^{n} \qquad (3.53)$$

where: $T_3 = \left[T_2 - T_1(K + Q^{n+1})\right]$ (3.54)

$$T_4 = \left[-T_2 - T_1(K - Q^{n})\right] \qquad (3.55)$$

T_3 and T_4 are tridiagonal matrices. Relation (3.53) gives the solution to beam advection in z direction. By giving a value for F^n at any point in radial direction, F^{n+1} is computed by using the Gaussian elimination technique [35] or cyclic reduction method [47], which will be discussed in Chapter 4.

CHAPTER FOUR
VECTORIZATION OF THE HEATER SUBROUTINE

One of the objectives in this project is to restructure the HEATER code, which will be implemented on the CRAY XMP and CYBER 205 supercomputers. The discussion on vectorization is therefore restricted only to these computers. Implicit vectorization is implemented in HEATER to ensure not only compatibility with the standard FORTRAN 77 but also portability to other environments. The numerical solution of a partial differential equation (ie. the paraxial wave equation 3.40) usually involves in the inversion of a tridiagonal matrix. Since the standard Gaussian elimination algorithm for solving tridiagonal equations is not vectorized, a new algorithm known as or the cyclic reduction method [47], is employed in HEATER. The revised HEATER code can run three to five times faster than the original on the CRAY-XMP and CYBER 205.

4.1 STRUCTURE OF THE HEATER CODE

HEATER is an independent subroutine which describes the propagation of a laser beam through a plasma. It can be ported into any two-dimensional Eulerian and Lagrangian hydrodynamic code in order to simulate the laser-plasma interaction processes. HEATER accepts input values from a hydrodynamic code for quantities such as electron and ion temperatures, ionization number, electron density, radial and axial hydrodynamic mesh, and laser-beam parameters. The laser-beam parameters, which are outlined in the XLAS1 array (Table 2), include

input power, spot size, wavelength, beam incoherency, focal point and pulse shape.

HEATER returns calculated values such as absorbed energy, laser intensity, and the radial and axial ponderomotive forces. These forces are important when considering self-focusing and filamentation processes, especially near the critical density. HEATER's structure can be summarized in Chart 3. From this flow chart, the routine can be divided into seven logical blocks, each of which can be described as follows.

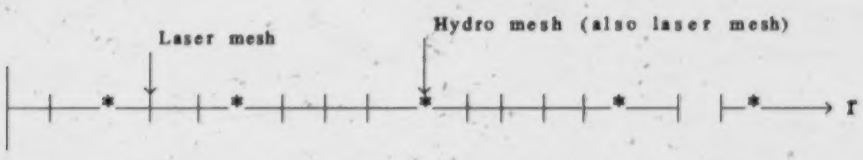
4.1.1 Block 1: Laser off.

In this Block, if the actual simulation time passed from the hydrodynamic code is greater than the time at which laser switched off, or if the energy delivered in the laser pulse is calculated to be greater than the available pulse energy, HEATER is switched off by assigning XLAS1(9) to -1.0. The absorption power, flux, and ponderomotive forces are initialized to zero before returning to hydrodynamic code (Chart 4).

4.1.2 Block 2: Laser on.

In this Block (Chart 5), if the logical value NLRES was true, and if the energy delivered up to current time is less than that in the laser pulse, and if the simulation time is less than the time in which laser is switched off, HEATER is turned on by assigning XLAS1(9) to 1.0. First, the radial beam profile is selected and necessary information is defined. The two types of laser beam profiles that can be specified are a Gaussian beam and a uniform beam superposed with a Gaussian perturbation. The spot size and the focal point are

established according to Verdeyen [56]. Next, the laser radial mesh is calculated by subdividing the hydro mesh. For a more accurate description of the beam, the laser radial mesh can be extended outside the plasma column to ensure that 99% of the total beam energy is accounted for. Once the laser radial points have been computed, the associated laser points are calculated so that they coincide with the hydro cells. The following picture clearly demonstrates the idea described above.

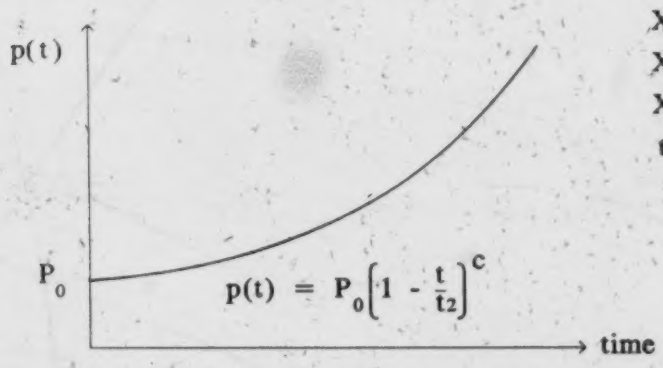


Once the laser grid has been constructed, the linear interpolation factors are calculated in order to compute the electron density, electron temperature, charge number and absorption coefficients at the laser mesh points. The laser position (measured form the axis) corresponding from lower hydro cell j to upper j+1 is computed. This is used to describe average laser quantities more accurately by integrating from the lower to the upper hydro cell. Third, the spline coefficients are calculated according to Fleck [49] to yield the entries for tridiagonal vectors T_1 and T_2. The mathematical derivation for vector T_1 and T_2 has been discussed in Chapter 3, Section 3.1. Once T_1 and T_2 vectors have been set up, a weighting factor is used in conjunction with the T_2 vector to attenuate the diffraction effect at the edge of plasma boundary. This factor results in reducing reflection outside the plasma grids.

4.1.3 Block 3: Input power.

In this Block (Chart 6), the laser input power is computed according to the type of input laser pulse. There are four kinds of laser pulses which are specified as follows.

1. Compression pulse: XLAS1(8) = 1

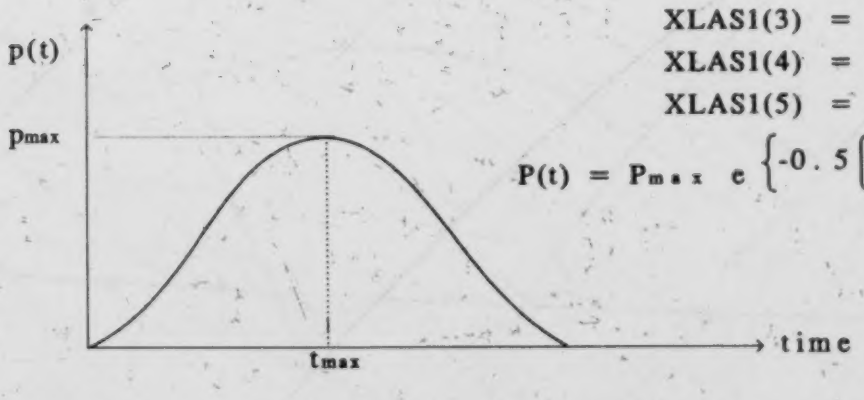


XLAS1(2) = t_2
XLAS1(3) = c
XLAS1(4) = P_0
t_2 = time laser switch off f

The power is expressed as:

$$p(t) = P_0\left[1 - \frac{t}{t_2}\right]^c \qquad (4.1)$$

2. Gaussian pulse: XLAS1(8) = 2



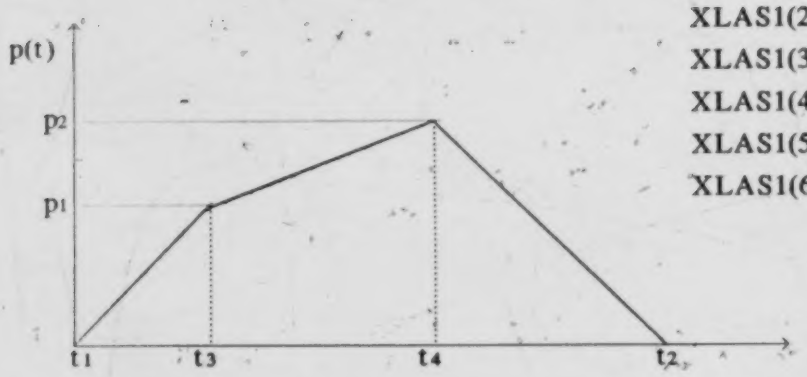
XLAS1(3) = t_{max}
XLAS1(4) = τ
XLAS1(5) = P_{max}

$$P(t) = P_{max}\ e^{\left\{-0.5\left[\frac{t - t_{max}}{\tau}\right]^2\right\}}$$

The power is expressed as:

$$p(t) = P_{max}\, e^{\left\{-0.5\left[\frac{t - t_{max}}{\tau}\right]^2\right\}} \qquad (4.2)$$

3. Triple ramp pulse: XLAS1(8) = 3



XLAS1(2) = t_2
XLAS1(3) = t_3
XLAS1(4) = t_4
XLAS1(5) = p_1
XLAS1(6) = p_2

for $t_1 < t < t_3$, then the power is expressed as:

$$p(t) = p_1 \left[\frac{t - t_1}{t_3 - t_1}\right] \qquad (4.3)$$

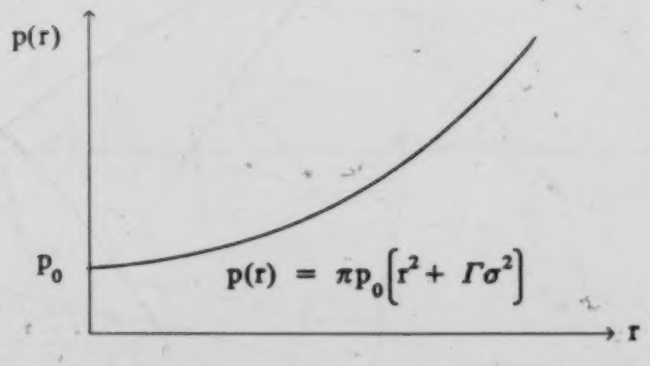
for $t_3 < t < t_4$, then

$$p(t) = \left[\frac{p_1(t_4 - t) + p_2(t - t_3)}{t_4 - t_3}\right] \qquad (4.4)$$

for $t_4 < t < t_2$, then

$$p(t) = p_2 \left[\frac{t_2 - t}{t_2 - t_4}\right] \qquad (4.5)$$

4. Plane wave with Gaussian perturbation pulse. XLAS1(8) = 4



XLAS1(5) = p_0

$$p(r) = \pi p_0 \left[r^2 + \Gamma \sigma^2 \right] \qquad (4.6)$$

where p_0 is the plane wave intensity

Γ is the fraction of plane wave intensity

σ is the beam spot size at the entrance plasma.

Once the pulse time profile (**1,2** or **3**) has been selected for given simulation, the input power is calculated accordingly for all value of time t. In example **4**, the power profile is a function of radial position only.

4.1.4 Block 4: Absorption coefficients.

In this Block (Chart 7), the absorption coefficients are calculated according to Dawson and Johnson [53]. The absorption coefficients are calculated on the hydro mesh and later used to extrapolate values at any laser grid points.

4.1.5 Block 5: Entrance plasma conditions.

In Block 5 (Chart 8), the complex flux amplitude at the entrance plasma is evaluated analytically. For the Gaussian pulse shape, the analytical solution for the complex flux amplitude is found to be [26,48].

$$F(r,0) = \left[\frac{p(t)}{\pi \sigma_0^2} \right]^{1/2} \frac{1}{C} e^{\left(\frac{-r^2}{2\sigma_0^2 C} \right)} \qquad (4.7)$$

where $C = 1 - i \frac{I \lambda Z_{foc}}{2\pi\sigma_0^2}$ (4.8)

I is the incoherency parameter

λ is the laser wavelength

Z_{foc} is the focal point in side plasma.

Equation (4.7) describes the complex flux radial profile based

upon the input power p(t). Once the complex flux amplitude at the entrance plasma is established, the Laplacian M(r,0) at the entrance plasma is easily computed by evaluating $\frac{1}{r}\frac{\partial}{\partial r}\left[r\frac{\partial F}{\partial r}\right]$, which is:

$$M(r,0) = F(r,0)\left[\frac{-2\sigma_0 C - r^2}{\sigma_0^2 C^2}\right] \qquad (4.9)$$

The electron density, absorption coefficients, electron temperature and ionization number are calculated at any laser point by extrapolating quantities defined on the hydro mesh. The Langdon correction factor [55] is calculated at every laser point in the radial direction. This correction factor accounts for the collective behavior of the electron distribution function. Langdon claims that, when $Zv_0^2/v_e^2 \geq 1$, the resulting non-Maxwellian velocity distribution can reduce absorption by a factor of up to 2 compared with the result for a Maxwellian distribution. Information about the beam at the entrance to the plasma is required in order to numerically evaluate beam advection in the z direction. In other words, it is a boundary condition which must first be known in order to compute the solutions at subsequent values of z.

4.1.6 Block 6: Solution at all radial and axial meshes.

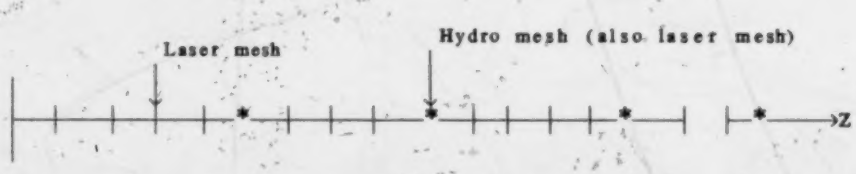
In this Block (Chart 9), the paraxial equation (3.25) is solved at any point in radial and axial grids within the plasma column. First, the acceptable step size Δz is chosen for which the following conditions [48] are imposed.

$$\int_z^{z+\Delta z} K_{abs}(r,z)dz \leq \frac{1}{10} \qquad (4.10)$$

where $K_{abs}(r,z)$ is assumed to vary linearly between mesh points and

$$\Delta z < \frac{1}{20} \left| \frac{1}{\sigma^2} \frac{d\sigma^2}{dz} \right|^{-1} \qquad (4.11)$$

These conditions ensure that no more than 10% of beam energy can be absorbed in one hydro step size and the change in the area of the laser spot should be less than 1/20 of the previous step. After the Δz has been chosen, the hydro grids can be subdivided into a finer laser mesh. The laser mesh is not allowed to be coarser than the hydro mesh. The use of a finer laser mesh can describe the beam behavior more accurately as it travels along the axial direction. However, more computations are required when a finer laser mesh is chosen. The laser mesh is chosen in such a way that one of the laser coordinates coincides with each of the hydrodynamic points (as in the case of radial subdivision). The following figure illustrates the idea described above.



The electron density, electron temperature, absorption coefficients and ionization number are extrapolated from the hydro grid and are used in the laser mesh at the current z plane. Q arrays are set up using relation (3.24). T_3 and T_4 matrices are constructed using relations (3.54) and (3.55), respectively. The complex flux for the next z plane (eq. 3.53) is solved by using the cyclic reduction method which will be discussed later. The Laplacian is found at the

next z plane by applying equation (3.44). The cyclic reduction method for solving tridiagonal matrix is also employed. The real time flux is calculated using relation (3.18). The absorbed power and the ponderomotive forces are computed by applying relation (3.39), (3.34) and (3.38) respectively. The flux, absorbed power and the ponderomotive forces are calculated on the laser mesh. Therefore, it is necessary to store all these values on the hydro mesh by averaging total quantities from total laser mesh within a hydro mesh. In order to save CPU time, the propagation of the laser beam is terminated when energy that has been lost (through absorption and ponderomotive work) is greater than 99% of the input energy at the entrance plasma.

A further consideration in HEATER is the rescaling of the beam energy to appropriate values due to the presence of ponderomotive forces which can either do work on, or extract energy from the plasma. The paraxial equation (Eq. 3.40) accounts only for absorption, diffraction and refraction effects, but it does not include ponderomotive work effects. Therefore, great care should be taken in cases where ponderomotive forces are present. These forces have a great influence on the beam energy profile which in turn may result in a density profile modification and filamentation of the laser beam. Therefore, the computed flux at every step has to be rescaled to account for the work done by ponderomotive forces. The beam rescaling corrects energy lost and ensures that energy conservation is maintained throughout the simulation process.

4.1.7 Block 7: Normalization of volume elements.

In Block 7 (Chart 10), the flux, absorbed power, axial and radial

ponderomotive forces are normalized and stored in variables defined on the hydro grid. The laser power escaping from the plasma is also computed. The relative errors, defined as a ratio of the input power minus the sum of absorbed energy, ponderomotive work and energy escape from plasma to the input power, are calculated at the end of HEATER subroutine.

4.2 IMPLEMENTATION OF HEATER ON THE CRAY XMP

The CRAY XMP vector computer is a register-to-register machine. Each vector processor has eight registers which act as very fast memories. Thus, the restructured HEATER code should have these registers being used effectively to optimize performance. The CRAY XMP compiler (CFT) vectorizes only the innermost DO loops. Therefore, HEATER is restructured in such a way that the innermost DO loops are the most efficient for vectorization. An objective of this project is to restructure HEATER in order to reduce computational time and increase performance. To implement HEATER on the CRAY XMP, most parts of HEATER, which are amendable to vectorization, have been restructured and rearranged according to the guidelines discussed in Chapter 2. Details of the vectorization can be seen in the HEATER program listing (end of this thesis). Only the major considerations in this vectorization processes are discussed below.

4.2.1 Avoiding assigned GO TO statements.

The unconditional GO TO statements within DO loops prevent the compiler from optimization. Therefore, the use of conditional IF and ELSE statements are employed throughout HEATER. The use of IF and ELSE

statements handling arrays in DO loops is not an obstacle to vectorization since the CFT compiler can generate a vector instruction code which is vectorizable. Consider the original part in HEATER which contains GO TO statements:

```
      IF (TIME .LT. XLAS11(1)) RETURN
      IF (TIME .GT. XLAS2(2)) GO TO 100
      IF (ELAS1 .GT. XLAS2(10)) GO TO 100
      IF (XLAS(9) .LT. 0.0 ) GO TO 200
      IF (NLRES) GO TO 200
```

To exercise the use of the IF then ELSE statement, the restructured code should be:

```
IF (TIME .LT. XLAS1(1)) RETURN
IF (TIME .LE. XLAS2(2)) THEN
      IF (ELAS1 .GT. XLAS2(10)) THEN
      IF (XLAS(9) .LT. 0.0 ) GO TO 200
      IF (NLRES) GO TO 200
      ENDIF
ENDIF.
```

Consider another example which contains the use of IF and ELSE statements to replace the use of GO TO statements:

```
      GO TO (410,420,430,440) IKIND
410   COMPRESSION PULSE
      .........
420   GAUSSIAN PULSE
      .........
430   TRIPLE RAMP PULSE
      .........
440   GAUSSIAN PULSE WITH PLANE WAVE PERTURBATION
      .........
```

The restructured code should be:

```
      IF (IKIND .EQ. 1) THEN
      COMPRESSION PULSE
      .........
      ELSEIF (IKIND .EQ. 2) THEN
      GAUSSIAN PULSE
      .........
      ELSEIF (IKIND .EQ. 3) THEN
      TRIPLE RAMP PULSE
      .........
      ELSEIF (IKIND .EQ. 4) THEN
```

```
      GAUSSIAN PULSE WITH PLANE WAVE PERTURBATION
      ....A....
      ENDIF.
```

In HEATER, most of the GO TO statements have been replaced by IF and ELSE statements (see program listing). The use of conditional IF and ELSE statements has an additional intrinsic advantage, in which the code is not only vectorizable but also structured.

4.2.2 Converting scalar variables into vector arrays.

The presence of vector arrays is a requirement for vectorization. The programmer should convert scalar variables into vector arrays or promote them into temporary vector arrays within a DO loop. In HEATER, most of scalar variables are converted into vector arrays. Associated with the conversion are advantages and disadvantages. The advantages are that the code is suitable for vectorization and the results can be stored into arrays and later reused. The disadvantages are the occupation of memory space and increasing memory traffic while loading and storing vector arrays. In HEATER, many temporary vector arrays have been used. They serve as working spaces to aid vectorization by storing intermediate results.

4.2.3 Unrolling short DO loops.

Some compilers are 'smart' enough to unroll short DO loops automatically. However, it is a good practice to unroll short DO loops to ensure vectorization. Consider the following example:

```
      DO 395 JL=IMESH1,IMESHX
      WRFRAC = REFRAC*(JL-IMESH) - 0.5
      WRFRAC = (2.*WRFRAC**2-1.5)*WRFRAC + 0.5
      DO 395 JD=1,3
      ZT2(JL,JD) = WRFRAC*ZT2(JL,JD)
395   CONTINUE
```

Since the inner loop has an index ranging only from 1 to 3, the

restructured loop should be unrolled the inner loop to increase performance.

```
      DO 395 JL=IMESH1,IMESHX
      WRFRAC = REFRAC*(JL-IMESH) - 0.5
      WRFRAC = (2.*WRFRAC**2-1.5)*WRFRAC + 0.5
      ZT2(JL,1) = WRFRAC*ZT2(JL,1)
      ZT2(JL,2) = WRFRAC*ZT2(JL,2)
      ZT2(JL,3) = WRFRAC*ZT2(JL,3)
395   CONTINUE
```

4.2.4 Referencing external functions within a loop.

The referencing external functions and subroutines within DO loops prevents vectorization. Consider the following example:

```
      C STATEMENT FUNCTION FOR LANGDON CORRECTION
      FLANGD(AL) = 1. - 0.553*AL/(0.375+AL)
      ..........................
      DO 1796 JL=1,IMESHX
      ZTE  = ....
      ZVG0 = ....
      ZZ   = ....
      WLANGD(JL) = FLANGD((0.464*ZZ*ZFLUX(JL)*XL1*XL1/
                  (ZVGO*ZTE))**.75)
1796  CONTINUE
```

The function FLANGD is short and can be expressed fully within a DO loop as:

```
      DO 890 JL=1,IMESHX
      ZARG=(XL1QF*ZAZS(JL)*ZFLUX(JL)/(ZVG0S(JL)*ZTES(JL)))**.75
      WLANGD(JL) = 1.0 - 0.553*ZARG/(0.375)+ZARG)
890   CONTINUE
```

The above DO loop is now vectorizable. When referencing external functions and calling subroutines within a DO loop, it is advisable to separate the vectorizable and non-vectorizable parts.

4.2.5 Avoiding repeated calculations.

One objective of this current research is to increase performance and reduce computational time when running a program on vector computers. One may expect that the repeated calculation results in

longer computational time and a substantial reduction in performance. In HEATER, many temporary vector arrays have been used to store the intermediate results to avoid repeated calculations. In other words, the results are calculated only once and stored for further use. For example, the values of interpolation factors ZHOR, ZHODR, the integral limit JMIN and JMAX from one hydro mesh to the next, and the laser mesh square ZRQS, electron temperature TE, absorption coefficients ZABS etc. are calculated only once and then stored into arrays. Whenever these values are needed, they can be loaded into memory. The drawback of having these temporary arrays are memory occupation and start-up time for loading and storing arrays back into memory. However, the use of temporary vector arrays in HEATER is considered only when the time loading and storing of a vector array into memory can be considered less than time for which the CPU instructs arithmetic operation. With long vector arrays and complex calculations, vector temporary arrays exhibit a large reduction in computational time and a substantial increase in performance.

Contrary to the above statements, storage that is no longer required should also be eliminated through the use of scalar temporaries. When results within a loop are intermediate values and only the last value may be important, a scalar temporary can be used to eliminate unnecessary storage. It results in less memory traffic, reducing the unnecessary storing time, space and increasing performance. The following formulas [37] can be adopted to ensure vectorizability in which:

scalar = scalar $\mp$ vectorizable expression

scalar = scalar * vectorizable expression

scalar = scalar / vectorizable expression

scalar = MAX(scalar,vectorizable expression)

scalar = MIN(scalar,vectorizable expression).

When expressed in this form, the scalar can be treated as a virtual vector array (vector promotion). Accordingly, the compiler then performs the calculations and stores only the final result into memory. The following example illustrates the procedure of introducing new vector arrays, temporary variables and vector promotion.

The original code is

```
      DO 710 JR=1,NRM1
      INDEX = IREPR(JR)
      DO 710 JS=1,INDEX
      IB = IBCR(JR) + JS
      IF (IB .GT. I4SIG .OR. ZFLUX(IB) .EQ. 0.) GO TO 715
      ZRHO = FLOAT(JS) / FLOAT(IREPR(JR))
      ZRHOD = 1 - ZRHO
      ZKA1 = (ZRHO*ZABS(JR+1,JZ)
           + ZRHOD*ZABS(JR,JZ))*WLANGD(IB)
      ZKA2 = (ZRHO*ZABS(JR+1,JZ+1)
           + ZRHOD*ZABS(JR,JZ+1))*WLANGD(IB)
      ZA1 = (ZKA2-ZKA1) / (ZKA1*ZDZC)
      IF(ZA1 .LT. 0.05*ZKA1) GO TO 702
      ZDZ0 = (SQRT(1. + 0.2*ZA1/ZKA1)-1.) / ZA1
      GO TO 706
  702 IF (ABS(ZA1) .GT. 0.05*ZKA1) GO TO 704
      ZDZ0 = 0.1 / ZKA1
      GO TO 706
  704 IF(ZA1 .LE. -5.*ZKA1) GO TO 710
      ZDZ0 = -(SQRT(1.+0.2*ZA1/ZKA1)+1.) / ZA1
  706 IF(ZDZ0 .LT. ZDZL) ZDZL = ZDZ0
  710 CONTINUE
  715 CONTINUE
```

The restructured code becomes:

```
      DO 712 IB=2,I4SIG
           ZKA1S(IB) = (ZRHOS(IB)*ZABS(JRP1S(IB),JZ)
     1     + ZRHODS(IB)*ZABS(JRS(IB),JZ))*WLANGD(IB)
           ZKA2S = (ZRHOS(IB)*ZABS(JRP1S(IB),JZ+1)
     1     + ZRHODS(IB)*ZABS(JRS(IB),JZ+1))*WLANGD(IB)
           ZA1S(IB) = (ZKA2S - ZKA1S(IB)) / (ZKA1S(IB)*ZDZC)
```

```
712  CONTINUE
     DO 714 IB=2,I4SIG
     IF (ZA1S(IB) .GE. 0.05*ZKA1S(IB)) THEN
      ZDZ0 = (SQRT(1.+0.2*ZA1S(IB)/ZKA1S(IB))-1.) / ZA1S(IB)
     ELSE
       IF (ABS(ZA1S(IB)) .LE. 0.05*ZKA1S(IB)) THEN
        ZDZ0 = 0.1 / ZKA1S(IB)
       ELSE
       IF (ZA1S(IB) .LE. -5.*ZKA1S(IB)) GO TO 714
        ZDZ0 = - SQRT(1.+0.2*ZA1S(IB)/ZKA1S(IB))+1.)/ZA1S(IB)
       ENDIF
     ENDIF
      ZDZL = AMIN1(ZDZL,ZDZ0)
714  CONTINUE
```

In the original code, the loop is not vectorizable due to the presence of the GO TO statements within the loop. The interpolation factor is calculated repeatedly every time the beam advects one axial step. The use of GO TO statement makes the code awkward and unstructured. The restructured DO loops are vectorizable and appear in a more elegant form than the original. This is accomplished by separating vectorizable and conditional vectorizable parts. The ZKA2S variable is a scalar variable and is promoted to a vector by the compiler during execution. Only the last value is stored into memory. ZRHOS and ZRHODS are interpolation factor arrays which are calculated only once in HEATER and are used whenever needed. The GO TO statements are replaced by the conditional IF and ELSE statements. The vector ZKA1 and ZA1S arrays are stored since they can be used in other loops.

4.2.6 Increasing chaining processes.

In register-to-register vector processors, the vector functional units can be chained together, thus allowing an overlap of related operations. In the CRAY XMP, a vector processor (1 pipeline) can chain two LOAD , one STORE, one ADD and one MULTIPLY together in one instruction cycle. The CRAY XMP also has eight vector registers which

act as a fast memory cache to hold intermediate results. This feature enhances performance because storing and fetching of data from memory are avoided. A rule of thumb used in vectorizing HEATER is to combine as many loops as possible. In this way, the compiler can easily instruct the use of vector registers more effectively. It also provides the programmer with more opportunity to eliminate unnecessary storage and common sub-expressions.

4.2.7 Using a new vectorizable algorithm.

A "good" vectorizable program does not suggest that only DO loops are vectorized. A good algorithm poorly coded is usually preferable to a poor one with optimal coding. Most of the vectorization guidelines from Chapter 2 have been considered in the restructuring of the HEATER subroutine. As a result, the most important consideration is the algebraic restructuring of the code which has appropriate new vectorizable algorithms. In the original code, the Gaussian elimination method algorithm has been used to solve the tridiagonal matrix systems. This algorithm has proven to be accurate and stable. However, it involves a recursive structure which is the most difficult task for vectorization and even impossible to vectorize implicitly. Consider a tridiagonal linear system of equations:

$$b_{i-1}x_{i-1} + a_i x_i + c_i x_{i+1} = y_i \qquad i = 1,2,\ldots,n.$$

with $b_0 = c_n = 0$. The standard Gaussian elimination algorithm for solving these equations is

$$d_i = (a_i - l_{i-1}c_{i-1})^{-1} \qquad i = 1,2,\ldots,n-1$$

and

$$h = b_i d_i \qquad i = 1,2,\ldots,n.$$

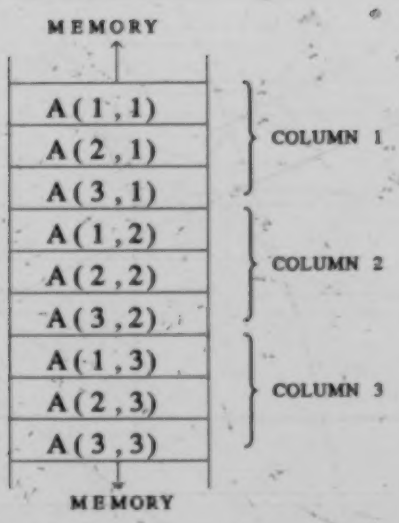
follow by

$$w_i = y_i - l_{i-1}w_{i-1} \qquad i = 1,2,\ldots,n.$$

Therefore, the solution is

$$x_i = d_i(w_i - c_i x_{i+1}) \qquad i = n,(n-1),\ldots,2,1.$$

The algorithm above is recursive since variables are coupled to one another. The recursive loop cannot be vectorized implicitly. Therefore, a new algorithm must be adapted to unfold uncoupling variables. A revised Gaussian elimination algorithm known as cyclic reduction [47] is suitable for a vector computer because of the uncoupling between variables and the elimination of recursiveness. This cyclic reduction algorithm is vectorizable and has been used to replace its previous algorithm used in HEATER. A detailed derivation of this algorithm is given in Appendix 3. To apply this method more effectively, the reduction step should be stored row-wise while the vector length column-wise. It implies that the vector length is coded as inner loops and reduction steps as outer loop because the characteristics memory layout treats multi-dimensional arrays as one large, stored column-wise in a single dimensional array. This feature enhances performance if the 'column vector' is longer than the row length. The following illustrates the physical storage of a two-dimensional array in memory.



Figs. 4.1 and 4.2 show the CRAY XMP and CYBER 205 benchmark of execution time versus vector length, respectively. The performance increases with an increasing vector length. The CYBER 205 shows nearly four times increase in performance for a vector length of 1000 elements, while the CRAY XMP is three times. With a vector length of 100 elements or less, the performance increase is a factor of 2 or less. This indicates that the performance increases as a function of vector length.

4.3 IMPLEMENTATION OF HEATER ON THE CYBER 205

In general, the implementation of HEATER on the CYBER 205 is similar to that on the CRAY XMP; however, some adjustments need to be made in order to improve performance. The CYBER 205 is a memory-to-memory vector computer and its compiler vectorizes only the contiguously stored vector data. Therefore, one needs not only to optimize the innermost DO loops but also to arrange the vector arrays

contiguously in the memory. In HEATER, most of variables have been stored in one-dimensional vectors by introducing temporary arrays. The CYBER 205 has a fixed start-up time that is independent from the vector length (Table 1). As a result, the performance increases in proportion to the vector length which can contain up to 2^{16} elements in total nested DO loops. This is a major drawback of the CYBER 205. In vectorizing HEATER, the procedures outlined in chapter 2 are employed.

The FT-200 compiler [46] cannot vectorize the conditional IF and ELSE statements within a DO loop. If these conditional statements are unavoidable, they are preferred to GO TO statements because the computer can generate a mask operation which handles the former more effectively than the latter. Unlike the CFT77 [45], the FT-200 compiler does not vectorize some intrinsic functions such as MAX, AMAX, MIN and AMIN. Therefore, all the loops containing these statements should be separated into vectorizable and non-vectorizable parts in order to increase performance.

In implementing HEATER, the link triad should be used as frequently as possible because this feature doubles the result rate (Table 1). The link triad allows the addition and multiplication vectors be linked together.

4.4 BENCH MARKING FOR THE CRAY XMP AND CYBER 205

In order to benchmark HEATER, a short code (Chart 1) is written to allow HEATER to run independently from the hydro code (ie. CASTOR). The purpose is to benchmark HEATER's performance for both the CRAY XMP

and CYBER 205 without involving the hydro code and to reduce computational cost. When bench marking HEATER, the scalar and vector mode are switched on and off. The resulting execution times are retained to determine the speedup rate by comparing the CPU time taken from each case. The CPU time taken in each case is measured as follows:

```
Start HEATER
Get CPU time1
     Execute Block 1
Get CPU time2
     Execute Block 2
          .
.....
Get CPU time7
     Execute Block 7
Get CPU time8
End HEATER
```

Each logical-block function in HEATER was described in Section 4.1. The total CPU time measured for both vector and scalar operations is calculated by subtracting the computational time from the previous one. For instance, the total CPU time used in HEATER is calculated as follow.

Total CPU time = time8 - time1.

The overhead time used to initialize variables can be eliminated by considering each individual block only. To get more accurate timing result for each machine, the same conditions can be run twice or more and then the average value is taken.

In comparing the speedup rate in HEATER for scalar and vector operations, graphical illustrations of execution time versus vector length (radial discretization) are presented. Fig. 4.3 shows the CRAY XMP's timing graph for vector and scalar mode and Fig. 4.4 describes the CYBER 205's, respectively. In both cases, the number of laser radial points is increased from 75 to 530 while the number of iterations (axial step) is fixed at 40. The speedup rate is improved by increasing the number of radial points implemented in inner DO loops. Among the logical blocks, Block 6, which solves for beam propagation, shows the most dominant performance factor. For Block 6, Fig. 4.5 and 4.6 illustrate the execution time versus number of radial points executed by the CRAY XMP and CYBER 205, respectively.

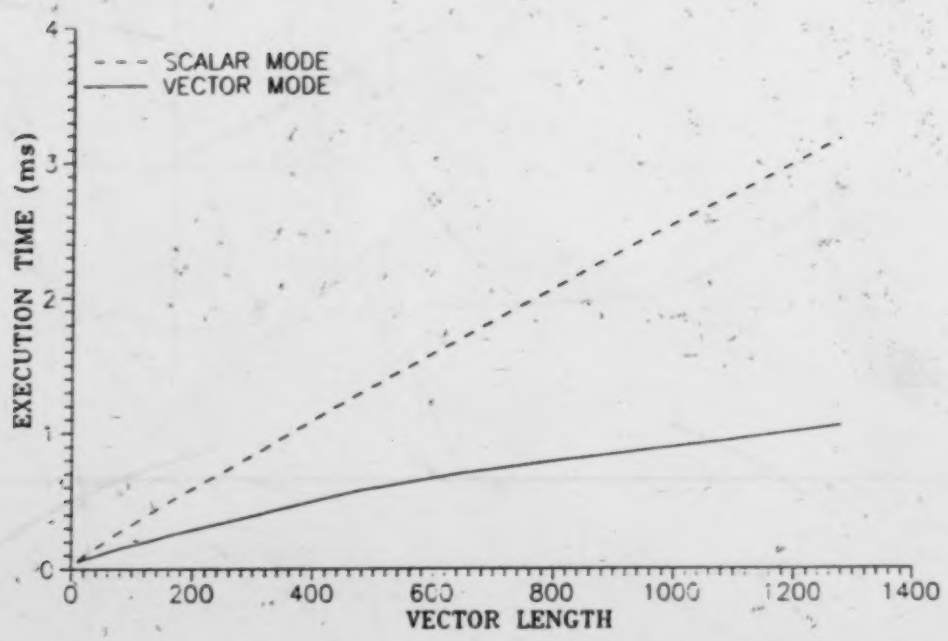


Fig. 4.1. Cyclic reduction performance comparision, CRAY XMP

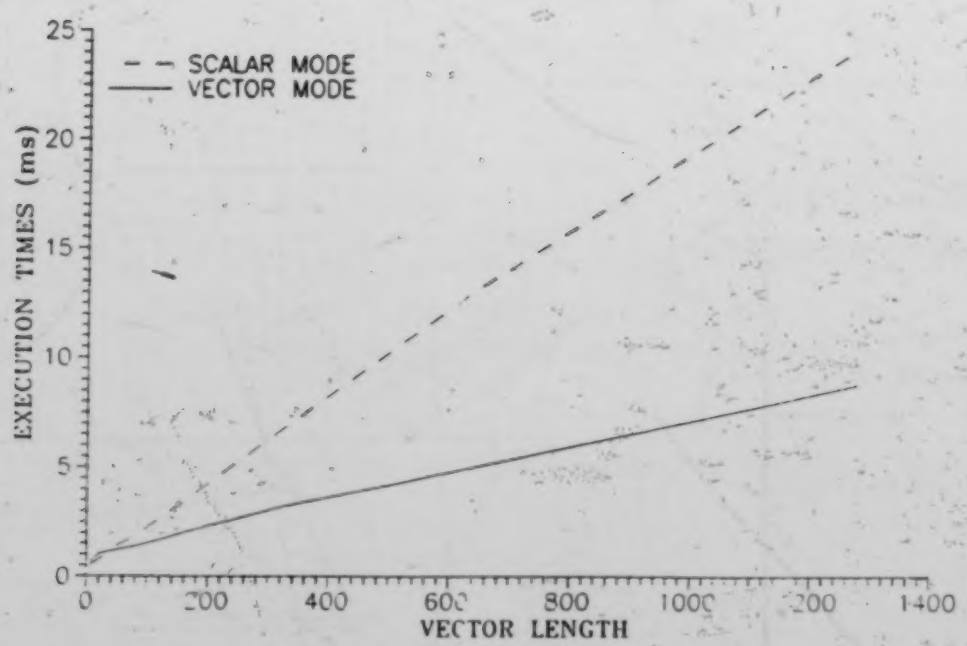


Fig. 4.2. Cyclic reduction performance comparision, CYBER 205

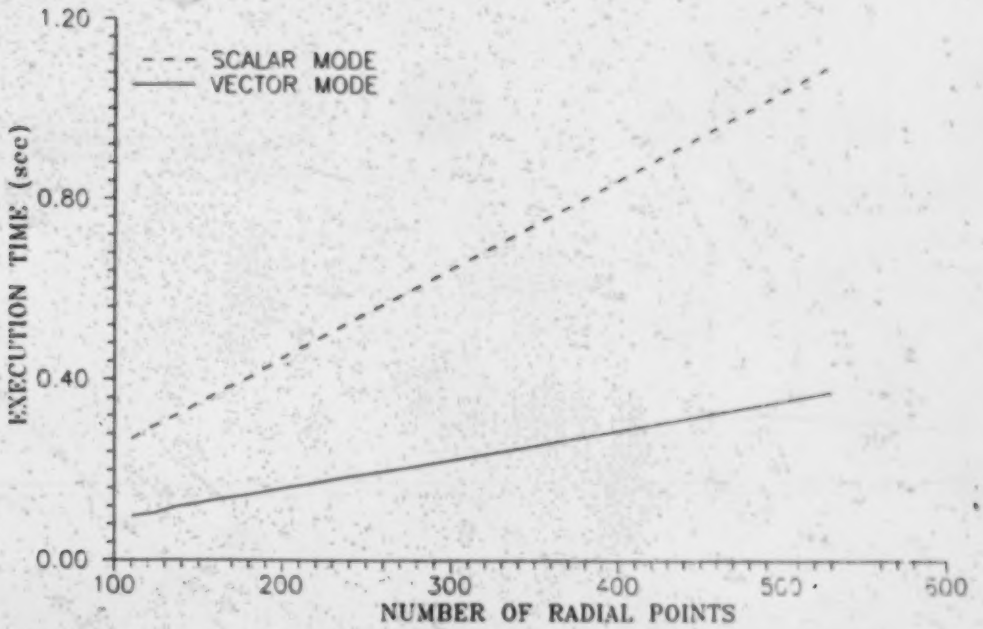


Fig. 4.3. Performance comparision of HEATER, CRAY XMP

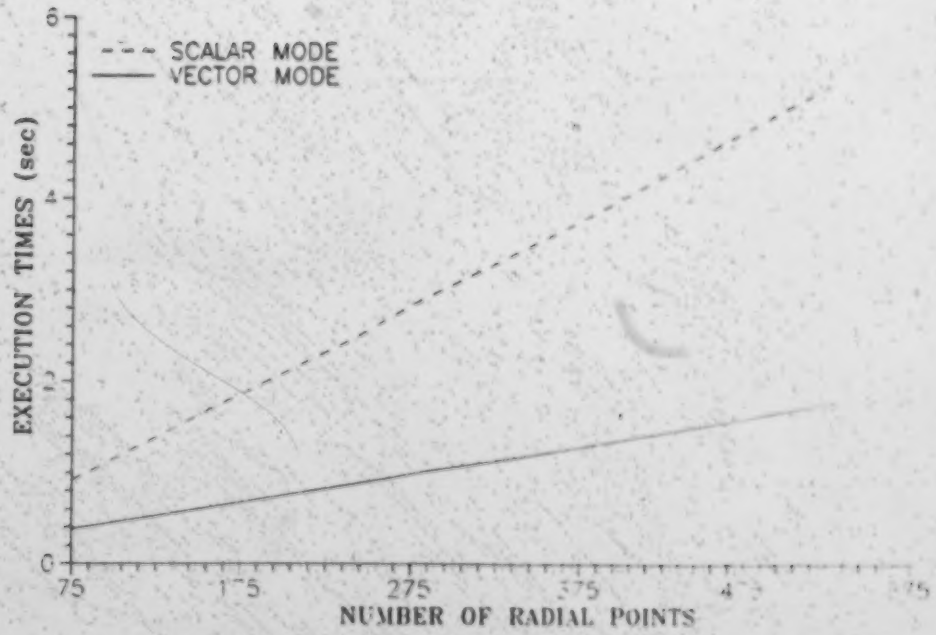


Fig. 4.4. Performance comparision of HEATER, CYBER 205

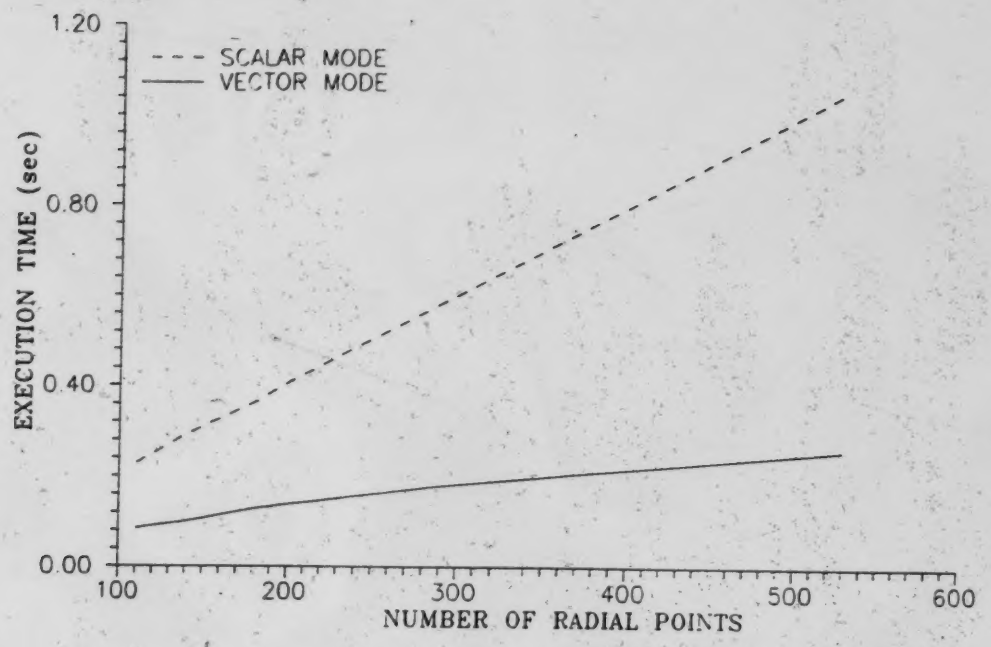


Fig. 4.5. Speed gain comparision of Block 6, CRAY XMP

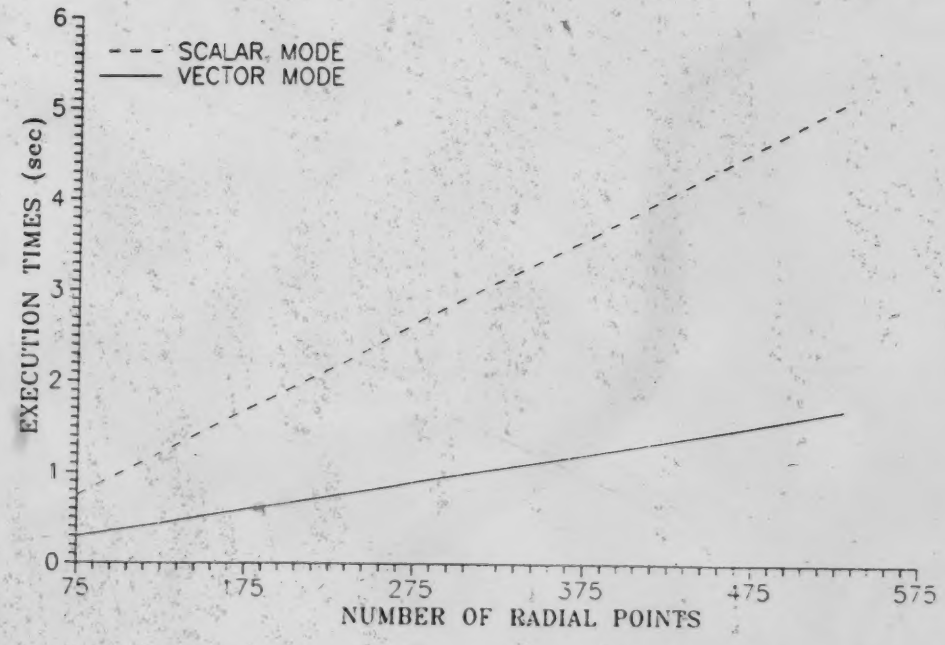


Fig. 4.6. Speed gain comparision of Block 6, CYBER 205

CHAPTER 5
COMPUTATIONAL RESULTS

The HEATER subroutine is an independent package which can be ported into any two-dimensional cylindrical symmetric Eulerian (ie. CASTOR [50,57]) and Lagragnian (ie. SALE [19]) hydrodynamic codes (MHD) to study laser propagation behavior and laser-plasma interaction processes. In this present research, HEATER has been interfaced with CASTOR to be part of a laser propagation module to simulate the time-dependent behavior of a laser-produced plasma. The examples presented in this thesis explore the effects with and without ponderomotive (PM) forces on the filamentation of laser light. The simulations take into account the time and spatial evolution of electron and ion temperatures, electron density, laser intensity, laser absorption, and radial and axial PM forces.

5.1 PRELIMINARY REMARKS FOR MHD CODE: CASTOR

5.1.1 Physical description.

The CASTOR code used at the University of Alberta is a two-dimensional, two-temperature MHD code which determines the dynamics of laser-produced plasmas. The simulation modeled is described in [50,57]. Briefly, this MHD code consists of an equation of state module, a diffusion equilibration module, a plasma-fluid advection module and a laser propagation module. The plasma is described by the dependent variables ρ, u, T_i and T_e which represent the density, velocity, ion and electron temperatures, respectively. A

number of options are available for the equation of state including the perfect gas law and a coronal equilibrium model.

5.1.2 Restriction on the complexity

The purpose for developing the CASTOR MHD code is to study and to better understand laser-plasma interactions. This code serves as a framework for further research involving laser-irradiated targets, laser-driven nuclear fusion and laser-produced X-ray lasers. Many MHD codes such as SALE, LASNEX, and SAGE have been developed to describe and explore the complexity of laser-plasma interaction processes. Each of the MHD codes described above has advantages and disadvantages. However, the overall combination of these codes has led to new insights into the physics and applications of laser-irradiated targets.

In CASTOR, the MHD equations are solved in a two-dimensional cylindrical (r,z) coordinate system with no ϕ dependence. The Eulerian coordinates are used. Therefore, the grid cells are first generated and fixed throughout the simulation. However, the physical description of laser-produced plasmas in the early stage is more accurately described by using Lagragnian coordinates since the mesh can move as plasmas expand. Energy transport by advection and diffusion is included in the code except that energy transport and losses through x-ray radiation have not yet been incorporated into the code.

5.1.3 Programming control and utilities.

The OLYMPUS [58] conventions (the control and utility package for initial value FORTRAN programs) are employed in the coding of CASTOR. This control utility program provides means to develop highly modular

structured codes in which extensions can be made, changed and debugged independently.

5.1.4 Relationship between HEATER and CASTOR.

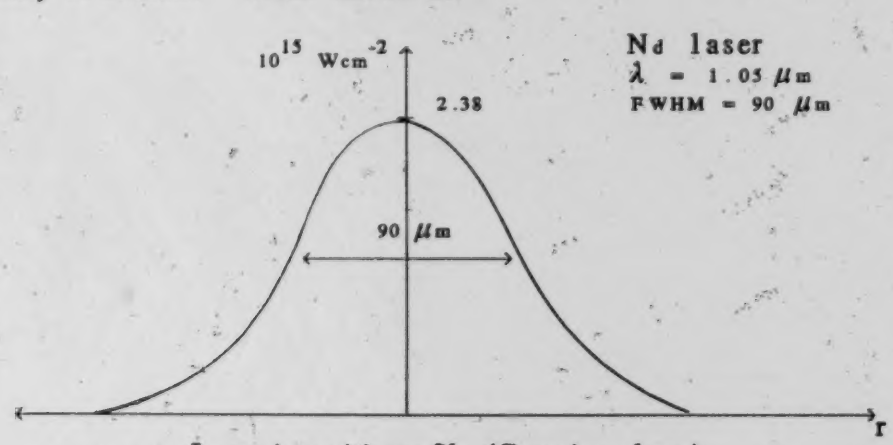
The HEATER subroutine is a laser propagation module in CASTOR. HEATER describes the beam propagation in the cylindrical plasma column assuming time independence and axial symmetry (no ϕ dependence). In HEATER, the laser electric field is approximated by solving the Helmholtz's wave equation. The direct solution to the wave equation results in the paraxial equation (Eq. 3.23). Diffraction, refraction and absorption effects are included in the paraxial equation which describe the spatial and time evolution of the laser beam. In addition, both radial and axial PM forces are also calculated in the HEATER subroutine. HEATER communicates with CASTOR through common blocks which are described in [50,57]. The main variables accepted from CASTOR are electron and ion temperatures, electron density, the charge number, and laser parameters which are outlined in XLAS1 array (Table 1). The paraxial wave equation is numerically solved by approximating the radial flux with cubic splines [49] and by using conservative spatial differencing (Crank-Nicolson) in the axial (z) direction.

The laser energy is deposited in each hydro cell through inverse bremsstrahlung. This in turn acts as a source term in the electron energy equation. The spatial mesh used in HEATER is obtained by subdividing the MHD mesh into a finer grid for increased resolution and accuracy. The results are then transformed back to the MHD mesh. The radial grid used in HEATER is described in Appendix 5. The radial

mesh spacings can be either uniform or non-uniform. Once chosen, the mesh spacing is fixed along z direction. In the axial direction, the mesh is allowed to change dynamically according to conditions (4.10) and (4.11) discussed in Chapter 4. During a time step, the laser grid adjusts itself according to the maximum beam width (typically four times the size of beam width). HEATER assumes the plasma parameters to be fixed during a time step so that the beam propagation equations are solved assuming a stationary medium.

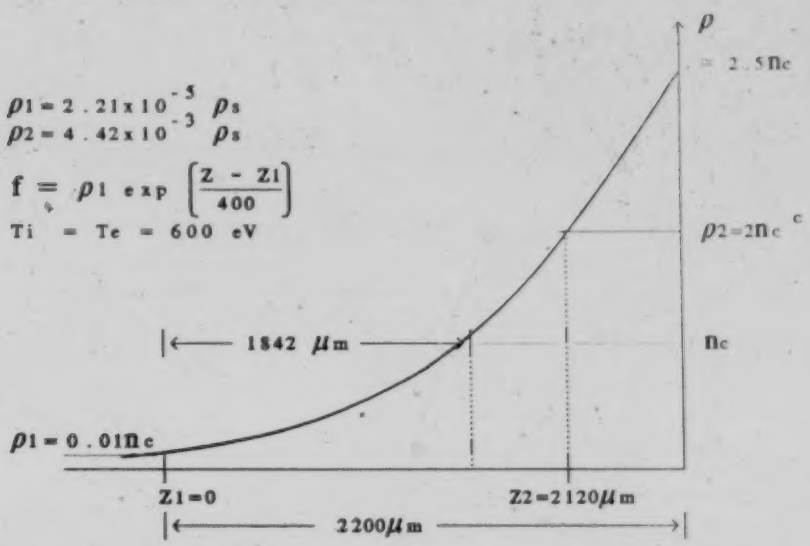
5.2 SIMULATION EXAMPLE

The use of HEATER with CASTOR will be illustrated by considering the irradiation of a Carbon target. The early phase of laser beam filamentation will be considered for the case where an intense laser beam is focused on a preformed Carbon plasma. The incident laser beam is assumed to be plane parallel wave with a constant incident intensity over a whole simulation process. The radial profile of incident beam is assumed to have a Gaussian profile with the on axis intensity of 2.38×10^{15} W/cm^2 shown as:



Laser intensity profile (Gaussian shape)

In this simulation, a Nd laser with a full width half maximum (FWHM) of 90 μm and with a 1.05 μm wavelength is used from which the critical density is calculated to be 1.1524×10^{27} m^{-3}. The assumed profile for the initial electron density is a double plateau connected with an exponential function. This profile is shown as follows:



Density profile: double plateau connect exponentially

The parameter ρ_1 gives an initial density which is 2.21×10^{-5} the solid density (ρ_s). The parameter ρ_2 gives a density which is 4.42×10^{-3} ρ_s. z_1 and z_2 are connected by the following function

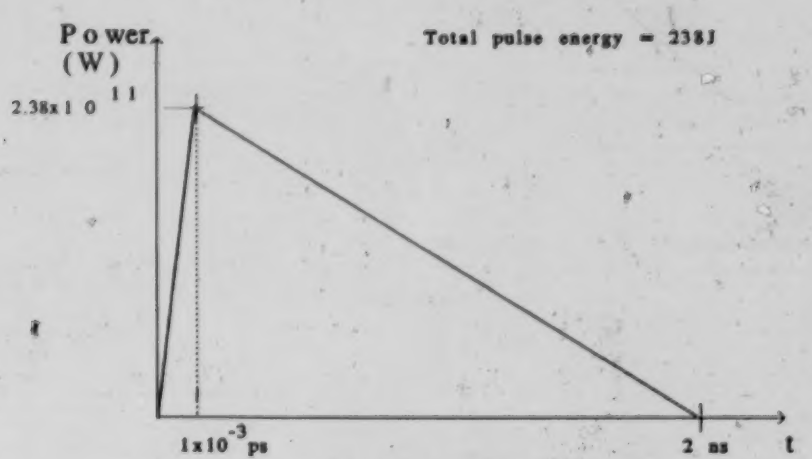
$$f = \rho_1 \exp\left[\frac{z - z_1}{p}\right],$$

where the plasma scale length p is calculated as:

$$p = \frac{(z_2 - z_1)}{\ln(\rho_2/\rho_1)} = 400\ \mu m.$$

In going from z_1 to z_2 with a plasma scale length of 400 μm, the density is changing from 0.01 of the critical density (n_c) to $2n_c$. The critical density n_c occurs at 2.21×10^{-3} ρ_s. For the assumed function

connecting between region z_1 and z_2, the critical density is located at 1842 μm away from z_1. The simulation of the hydrodynamic behavior of the plasma is made using a non-uniform 50x50 grid to represent a spatial region of 300 μm by 2200 μm in the radial and axial direction, respectively. The initial electron and ion temperatures are assumed uniform and have a value of 6.96×10^6 K (600 eV). The minimum time required to form noticeable filamentation is about 10 ps and the total simulation time is 30 ps for which 60 time steps are required. The minimum time interval between each step is 2.5×10^{-14} s. The computed data are stored on a magnetic tape at every 10^{th} step and can later be retrieved to produce 2D and 3D plots. The total pulse energy is about 238 Joules; however, only a portion of it is required to trigger the early phase of filamentation. The pulse has a triangular profile with its peak power density of 2.38×10^{11} Watts at time $\approx 1.0 \times 10^{-3}$ ps and linearly decreases to zero after 2.0 ns. The profile is shown as:



Power density profile: triangular pulse

The preformed plasma condition described above is a warm and long scale length plasma. This initial condition allows the beam to easily

propagate down the plasma column and to penetrate to a region near the critical density. In the front-end of the plasma column, where the density is low, the laser beam experiences little absorption and PM effects, thus allowing the beam to propagate further. Near the critical density region, the laser absorption is high, the PM forces are dominant, the laser intensity profile is altered, which in turn affects the electron density profile. These effects are discussed in the following sections through the presentation of 2D and 3D plots.

5.3 RESULTS: THREE-DIMENSIONAL REPRESENTATION

The simulation results can be best described by 3D plots of electron and ion temperatures, plasma density, laser absorption, laser intensity, radial and axial PM forces.

5.3.1 Electron temperature profiles.

Fig. (5.3.1) shows the time and spatial distribution of electron temperature. It is clear that the electron temperature is higher in the vicinity of the laser beam axis than elsewhere. The electron temperature then gradually decreases to its initial value when reaching the edge of plasma column. In the axial direction, the electron temperature increases gradually to its peak value at the focusing point near the critical density region, and then falls back rapidly to its initial value. Qualitatively, this phenomenon indicates that the laser can only penetrate the plasma up to the critical density. Near the critical region, high absorption of the normal incident light gives rise to a high plasma temperature.

5.3.2 Ion temperature profiles.

The ion temperature profiles (Fig. 5.3.2) are more uniform than those of the electrons. These figures indicate that very little ion heating has occurred. The absorption of laser energy is mainly through the classical inverse bremsstrahlung process which results in electron heating only. The energy absorbed by electrons is then transferred to the ions through collisional processes. Due to the fact that ions are much heavier than electrons, little increase in ion temperature has occurred as shown in Fig. (5.3.2). Near the critical region, the ion temperature shows an enhancement in the electron-ion collisional process. As a result, there is only slight perturbation in the ion temperature profile with a slightly higher value than in surrounding regions. Outside the focal region, both the electron and ion temperatures are in equilibrium.

5.3.3 Absorption profiles.

The laser absorption profiles are similar to the electron temperature profiles. Since the absorption is mainly through inverse bremsstrahlung, a higher energy absorption results in a higher electron temperature. In Fig. (5.3.3), the laser absorption increases as the electron density increases and is proportional to the laser intensity (Eq. 3.40). In the front-end plasma, little absorption has occurred, thus allowing the beam to travel further into plasma. The absorption value increases to its maximum near the axis of propagation and near to the critical density region.

5.3.4 Ponderomotive force profiles.

There is an interaction between the laser beam and the plasma

through the PM forces. These forces can greatly influence the physical behavior of the plasma and therefore beam propagation. The PM forces are proportional to both electric field intensity and its gradient (shown in Eqs. 3.26 and 3.27). Both the time and spatial evolution of radial and axial PM forces are shown in Figs. 5.3.4 and 5.3.5, respectively. These forces (especially in the radial direction) are shown to have a noticeable effect in the region near to the critical density where the laser intensity is the highest. In the front-end of the plasma, the PM forces are weaker than when they are near the focusing plane in the vicinity of the critical region. The radial PM force appears to have strong effects in altering the laser intensity and the electron density profile (shown latter in 2D plots). This force causes the electrons to be driven away from the high electric field intensity and the steep gradient region. As a result, the electric field experiences pulling forces, which eventually lead to the beam breaking up (filamentary or multi-filamentary structures).

One effect to consider for laser-plasma interaction is the beam energy being transferred to or extracted from plasma by PM forces. If the resultant PM forces and the fluid velocity are co-linear, the work done is positive, which implies that the laser beam has transferred energy to plasma. If the resultant PM forces and the plasma fluid velocity are not co-linear, the laser beam has acquired energy from plasma. Energy gain and lost through PM effects can degrade the beam structure, thus causing the beam to break up into multi-filamentary structure.

5.3.5 Electron density profiles.

Near the critical density region, the modification of the electron density profile by an intense laser beam is an important factor in the laser-plasma interaction process. This occurs as a result of PM and thermal forces acting on a plasma. The structure that is created by the PM and thermal kinetic forces interacting with plasma can in turn modify the propagation of the laser beam. Consequently, the plasma density profiles can further be modified. An example of such a feedback processes is illustrated in Fig. 5.3.6. The plasma density is essentially unchanged outside the laser focal spot because the PM forces are calculated to be weak and the laser absorption is low. Near the critical density region and on the propagation axis, the beam intensity is maximized (beam swelling) due to the fact that the group velocity becomes small. The PM forces are strongest here and thus gives rise to a beam being further self-focused. This is the main factor causing the beam filamentation and density depletion.

One of the mechanism causing this modification to the plasma profile is the momentum deposition by normally incident light near the plasma's critical density [51]. The PM forces push electrons away from the region of high laser intensity causing the plasma density to be lower than that of the surrounding region. Other complicated mechanisms affecting the density (not included in the model) are parametric and back scattering instabilities [51,52,57]. The parametric instabilities result from the interaction between the normal incident plane wave and the density perturbations. The back

scattering instabilities occur when the PM forces interact parallel to the direction of wave vector. Examples include stimulated Raman scattering and stimulated Brillouin scattering [52]. These nonlinear instabilities in the underdense plasma are not discussed further and they have been mentioned just to indicate that there are many other processes which can affect laser-plasma coupling.

5.3.6 Laser flux profiles.

The filamentation process can have an important effect in the coupling between laser light and the plasma medium. In the above discussion, the instabilities can alter the beam structure from an initial Gaussian profile into a filamentary (or multi-filamentary) structure as shown in Fig. 5.3.7. In the case of long scale length plasma, the instability can occur throughout the underdense region. Both filamentation and beam self-focusing are driven by either PM or thermal kinetic forces, or both. The onset of filamentation in this simulation is mainly driven by the PM forces near the critical density region. Three-dimensional plots in Fig. 5.3.7.b show the development of laser flux into multi-filamentary structures at time t = 9.88 ps. At this time, the electron density profile deviates slightly (2D Figs. 5.4.5.a and 5.5.6.a) from its initial profile. At time t = 18 ps, the laser flux has completely evolved into a filamentation structure (Fig. 5.3.7.c). The hot spot on axis near the critical density has an intensity of 25×10^{21} W/cm^2 while background intensity is essentially negligible (10^{17}W/cm^2). The density in the vicinity of laser hot spot has been depleted (Fig. 5.3.6.c and 5.4.6) to a much lower value. In this simulation, the onset of filamentation and the density depletion

are greatly influenced by the PM forces, especially in the radial direction. In practice, such intensities would not be obtained since other processes such as parametric instabilities would prevent this from happening.

5.3.7 Laser and plasma profiles without ponderomotive forces.

When PM forces are suppressed, the laser-plasma interaction occurs only through thermal kinetic forces. The electron and ion temperature profiles differ from those found when PM are on. The electron density and the laser flux profiles show a large discrepancy.

The electron temperature profile (Fig. 5.3.8) indicates a much hotter plasma on the axis than in the previous case (3103 eV in Fig. 5.3.8.a as compared with 1988 eV in Fig. 5.3.1.c). The temperature is also seen to decrease more rapidly in the radial direction. In the beginning, a hotter plasma on the axis and a colder plasma away from its axis affect directly the absorption coefficients (Eq. 3.22). As a result, the absorption coefficients are lower near the center and higher in the region away from the axis of propagation. Consequently, the absorption profile of a laser beam has a tendency to shift away from the axis of propagation as time evolve (Fig. 5.3.10). Similar to the case when PM force is on, the ion temperature profiles (Fig. 5.3.9) show a little change with the exception that a slight disturbance is exhibited in the region near to the critical density.

As a result from the above discussion, the beam tends to self-focus and develop into multi-filamentary structures (Fig. 5.3.12) away from the axis. The filamentation process driven by thermal forces (PM forces are turned off) is slower than that observed in the

simulation where PM forces are included. When PM forces are included in the simulation, the filamentation can be detected as early as 10 ps (Fig. 5.3.7.b); otherwise, the multi-filamentary structure occurs at about 30 ps (Fig. 5.3.12.c).

In general, the density depression is shown to have little effect when PM forces are not present. In the previous discussions, the PM forces have the strongest influence on the plasma near the focusing plane and on the axis of propagation, where the electric field pressure is the strongest. When the PM forces are turned off and the beam is being self-focused away from the propagation axis, the filamentation occurs away from the axis and the electron density experiences only a strong thermal force. As a result, the density structure has also steepened slightly away from the axis as shown in Fig. 5.3.11.

5.4 RESULTS: TWO-DIMENSIONAL REPRESENTATION

To better understand the laser-plasma interaction processes, the temporal variation of laser flux and plasma parameters are presented in 2D plots. There are several profiles which show the effects of laser and plasma parameters in the radial direction only. The evolution with time and axial distance are specified in the legend. The typical axial positions chosen at z = 1.5 and 1.7 mm represent 0.5 and 0.75 of the critical density, respectively.

5.4.1 Laser intensity profiles.

Fig 5.4.1.a and 5.4.2.a show the evolution in time of the laser intensity profile at z equal to 1.5 and 1.7 mm, respectively. Fig.

5.4.1.a shows that a multi-filamentary structure can be detected at as early as 10 ps. The beam shape strongly deviates from its initial Gaussian shape at the entrance to the plasma. The radial profile (Fig. 5.4.2.a) shows a strong beam focusing on the propagation axis. The on axis electric field intensity reaches a value of 500 times larger than its adjacent intensity region.

When the PM force is suppressed, the multi-filamentary structure develops later as shown in Figs. 5.4.1.b and 5.4.2.b. At 10 ps, the electric field indicates only a slight perturbation in the spatial profiles. At 20 ps, the laser filamentation starts to form. The beam tends to focus away from the propagation axis.

5.4.2 Ponderomotive force profiles.

The axial PM forces shown in Fig. 5.4.3.b and 5.4.4.b are small compared to their radial counterparts in Fig. 5.4.3.a and 5.4.4.a. The radial PM forces have an oscillatory profile which plays a major role in seeding filamentation and self-focusing. These forces are strongest where the electric field is the most intense and has the steepest gradient. The positive and negative parts of these forces when pushing electrons away from the region creates density depletion, and when attracting electrons raises their electron density. As a result, the density profile fluctuates as seen in Fig. 5.4.5.a and 5.4.6.a.

5.4.3 Electron density profiles.

The effects of PM forces play an interesting role in altering the electron density profile. In the above discussion, the electrons experience forces which push them away from the region of high electric field pressure. Consequently, the density has been steepened.

The profiles shown in Fig. 5.4.5.a and 5.4.6.a clearly illustrate the idea of density depletion. At a time of 10 ps, there is a slight perturbation in the electron density. At a time of 18 ps, the density fluctuation has become much larger.

When the PM force is not considered in the simulation, density fluctuations take a longer time to form. There is a slight perturbation in electron density as compared with the case when the PM force is present (Figs. 5.4.5.b and 5.4.6.b). These illustrate that the presence of PM force greatly enhances electron density perturbations.

5.5 SELF-FOCUSING AND FILAMENTATION OF LASER LIGHT

Self-focusing and filamentation of laser light is important to understand laser initiated fusion and other applications such as X-ray lasers, plasma diagnostics and laser-material processing. The processes that can cause a laser beam to self-focus include the radiation pressure, PM forces and thermal heating. Each can contribute to producing a density depression. The other physical effect affecting the defocusing of the beam is diffraction. The combination of both self-focusing (filamentation) and defocusing are the dominant effects that determine the nature of beam propagation.

5.5.1 Physical effects on self-focusing and filamentation.

R.D. Jone et al, R.S. Craxton and R.L. McCrory [15,17] have found that self-focusing behavior is very dependent on the detailed characteristic of the laser beam and plasma conditions. The simulation in the previous section has proven this effect. Generally, with short

spatial scale length plasma, ponderomotive filamentation is a dominant effect while for a larger scale length plasma, the thermal self-focusing is usually the dominant mechanism [15].

The absorption of laser energy also plays an important effect on self-focusing and filamentation of laser light. As a conclusion from the above simulation, any change in the intensity brought about by absorption will change the plasma parameters, which in turn affect the absorption. As a result, a multiple filamentary structure can form as shown in the previous section. In the case of the long low-density plasma column the focusing and defocusing of a laser beam has been observed [7,10,11,28,24].

Ponderomotive and thermal forces effects are the main factors causing the beam to focus and defocus. Ponderomotive filamentation is found to be important for small scale hot spots, while the thermal self-focusing is important in the focusing of a whole beam and large scale hot spots [15,17]. This simulation has clarified the dominant effects of PM force in the filamentation of laser light, especially near the focal region and close to the critical density region.

5.5.2 Applications and effects of filamentation.

The focusing of the laser beam (filamentation) causes the intensity to increase. Consequently, the PM forces become important in these high intensity regions causing the depletion of electron density profiles and the deviation of laser intensity profiles. The filamentation of laser light can drive the plasma to be unstable and causes a non-uniformity in both the electric field pressure and the density structure. These effects are undesirable when considering

nuclear fusion processes and the creation of long plasma columns for X-ray laser production.

When considering X-ray lasers, filamentation and self-focusing also play an important role in determining the efficiency and coherency of X-ray lasers. C J Keane et al., Richard London et al., C L Lewis et al. and S. Suckewer [28,31-34] have discussed the use of an optical laser as a pumping agent to produce soft X-ray lasers. According to their description for X-ray laser design, one or more optical lasers are employed to produce X-ray lasing spectra. First, a relatively low power (high energy) focusing laser beam is irradiated on a thin foil target to produce a long cylindrical plasma column. A second laser with extremely high power, short time duration is used to further ionize the plasma, creating the necessary population inversion conditions. As a result, X-ray radiation is emitted from each end of a plasma column. The physics of lasers and the principles of optical design are not mentioned here and can be found in most texts on the subject and in journal publications. Qualitatively, when laser interacts with plasma, the filamentation of the beam can spoil the uniformity of a long plasma column and the population inversion conditions, which in turn degrade the efficiency of X-ray laser gain medium.

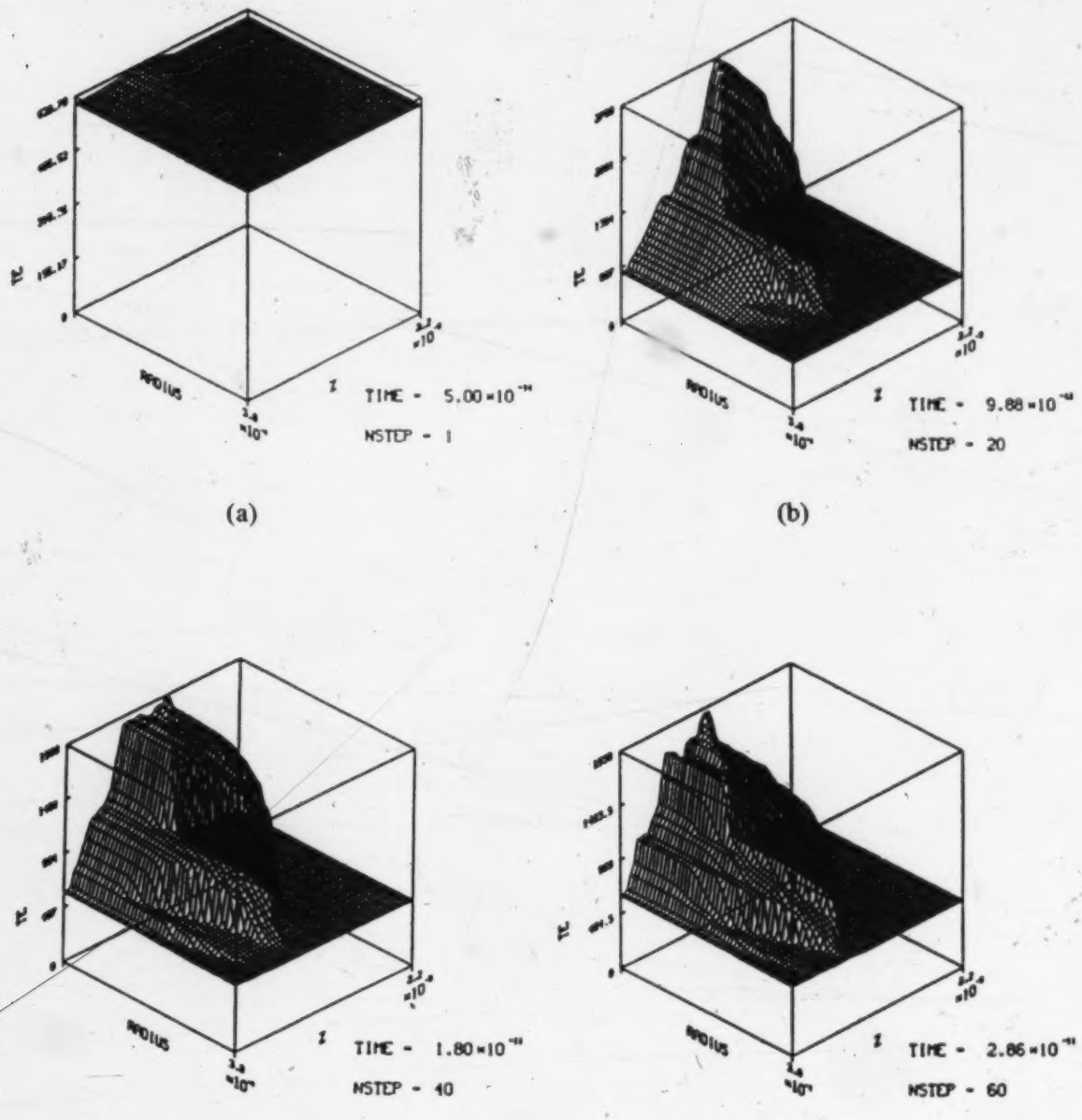


Fig. 5.3.1 Time and spatial distribution of electron temperature (PM on).

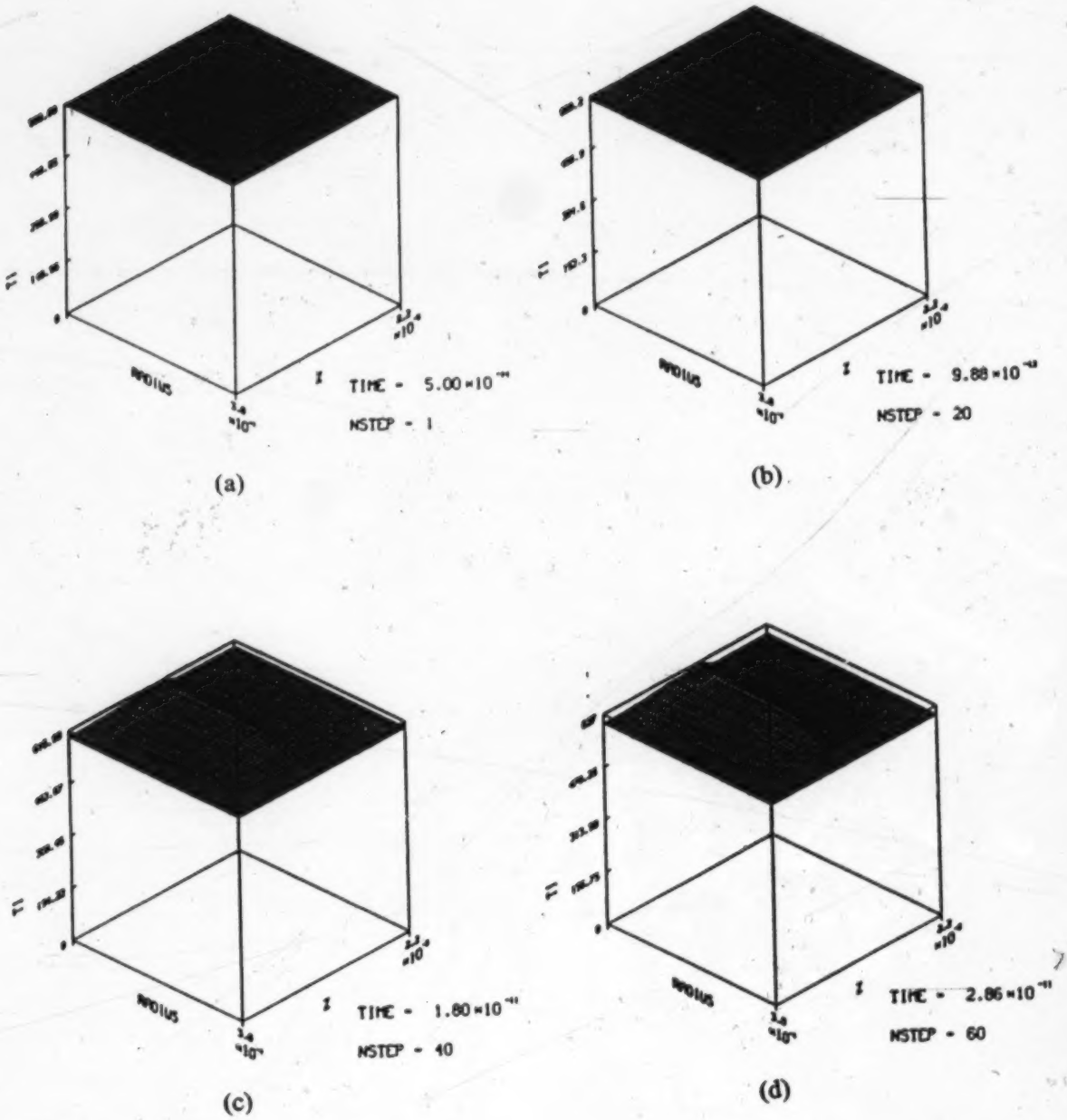


Fig. 5.3.2 Time and spatial distribution of ion temperature (PM on).

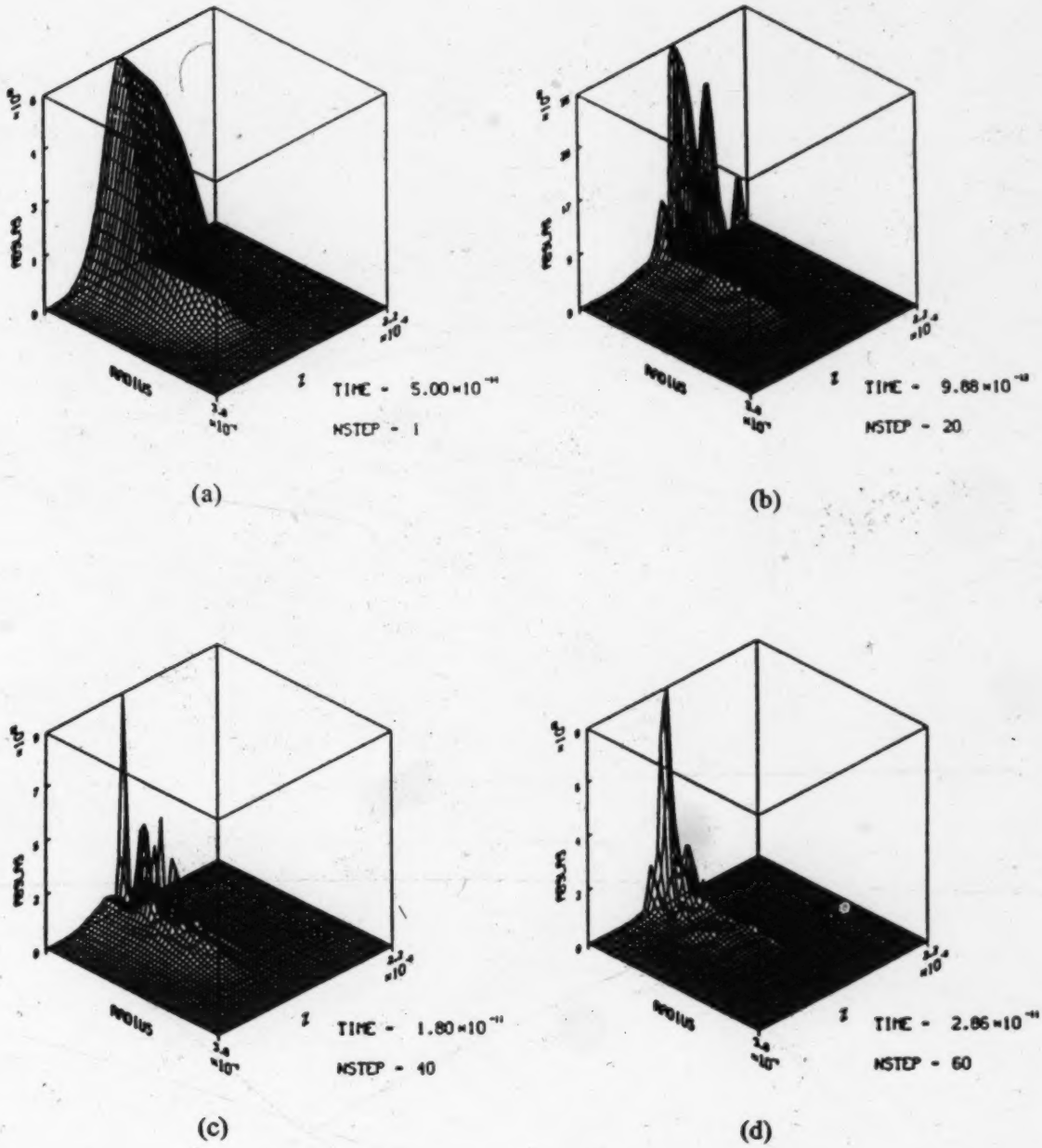


Fig. 5.3.3 Time and spatial distribution of laser absorption (PM on).

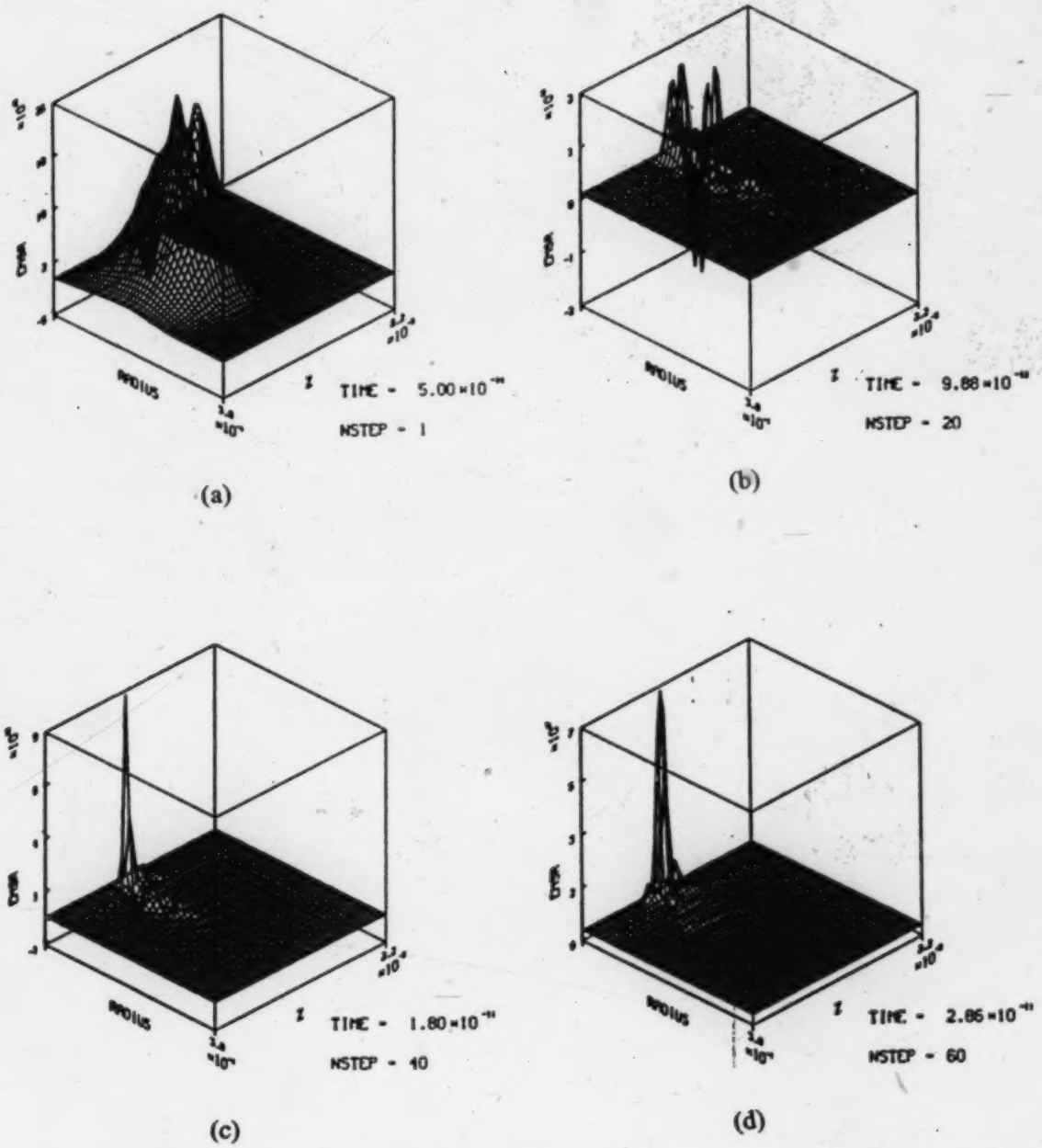


Fig. 5.3.4 Time and spatial distribution of radial ponderomotive force.

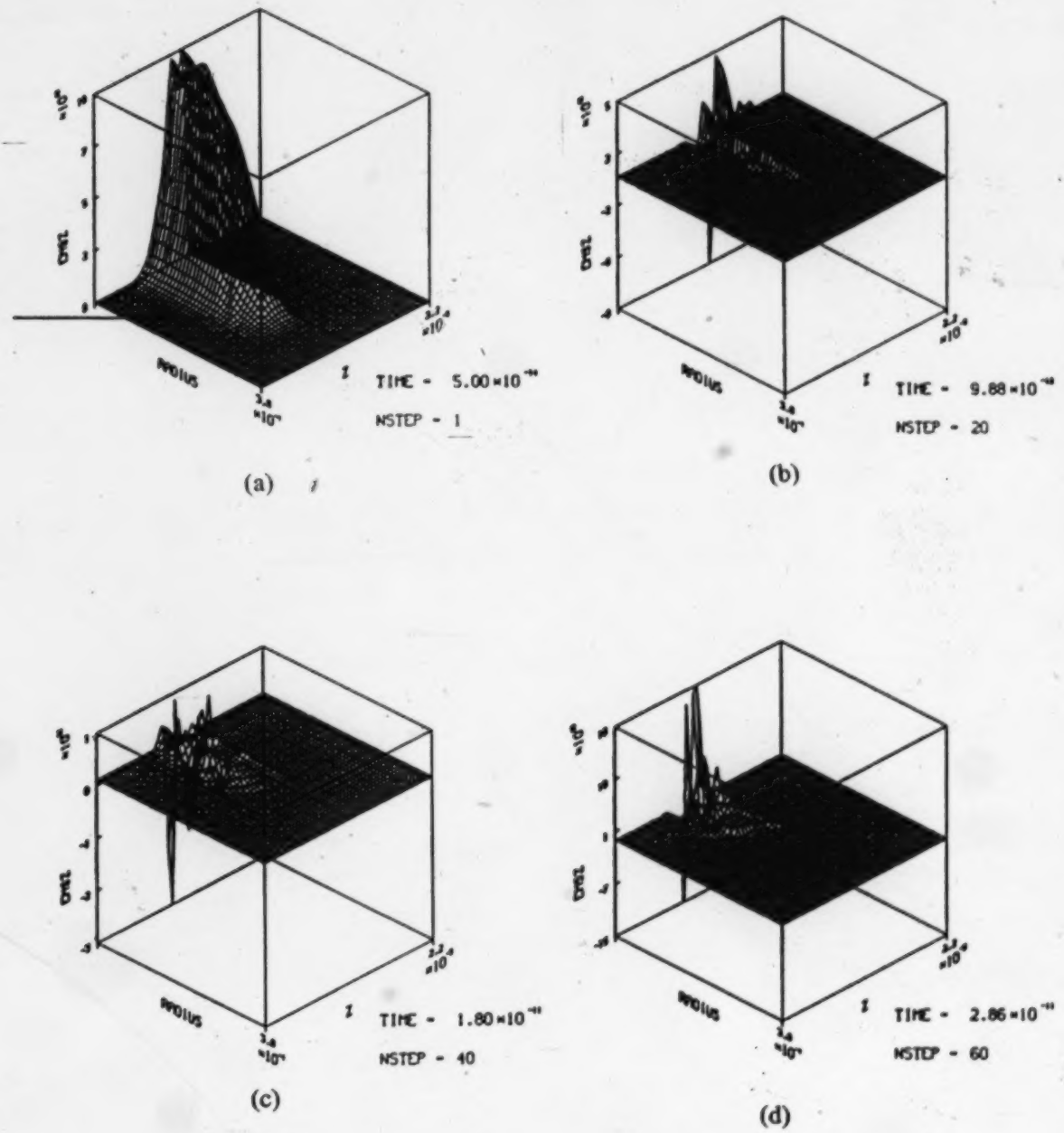


Fig. 5.3.5 Time and spatial distribution of axial ponderomotive force.

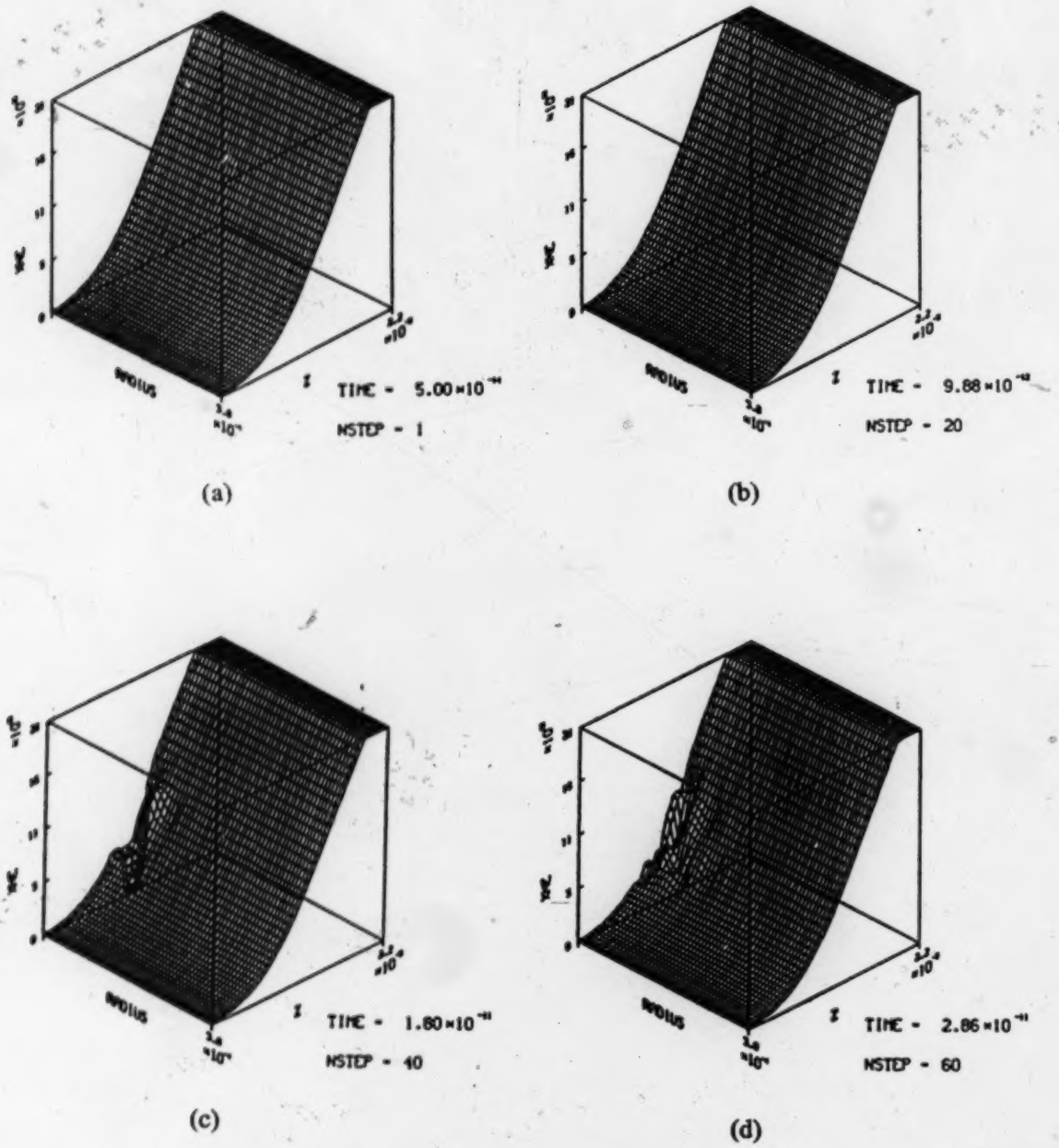


Fig. 5.3.6 Time and spatial distribution of electron density (PM on).

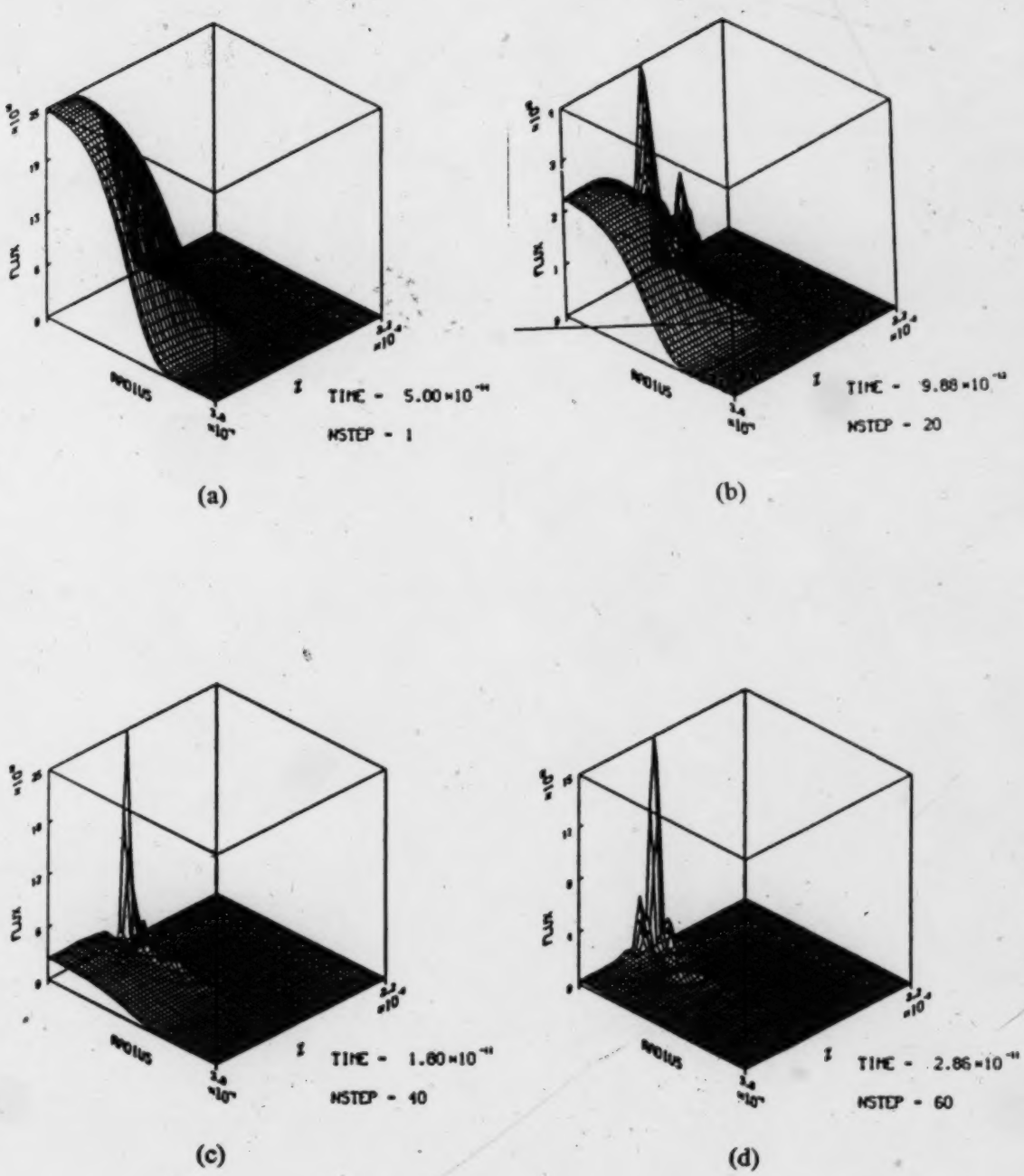
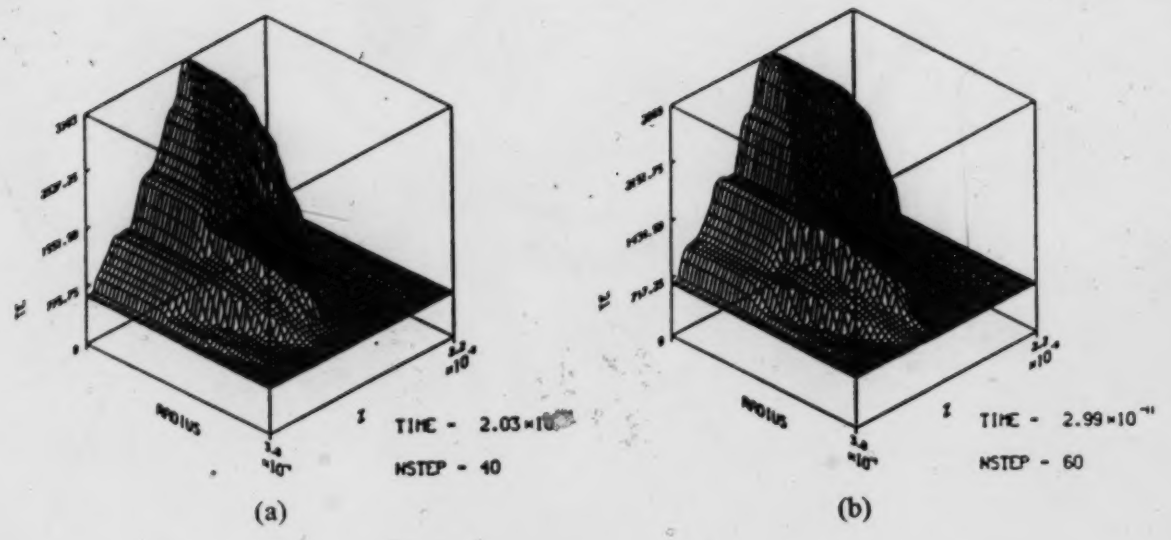
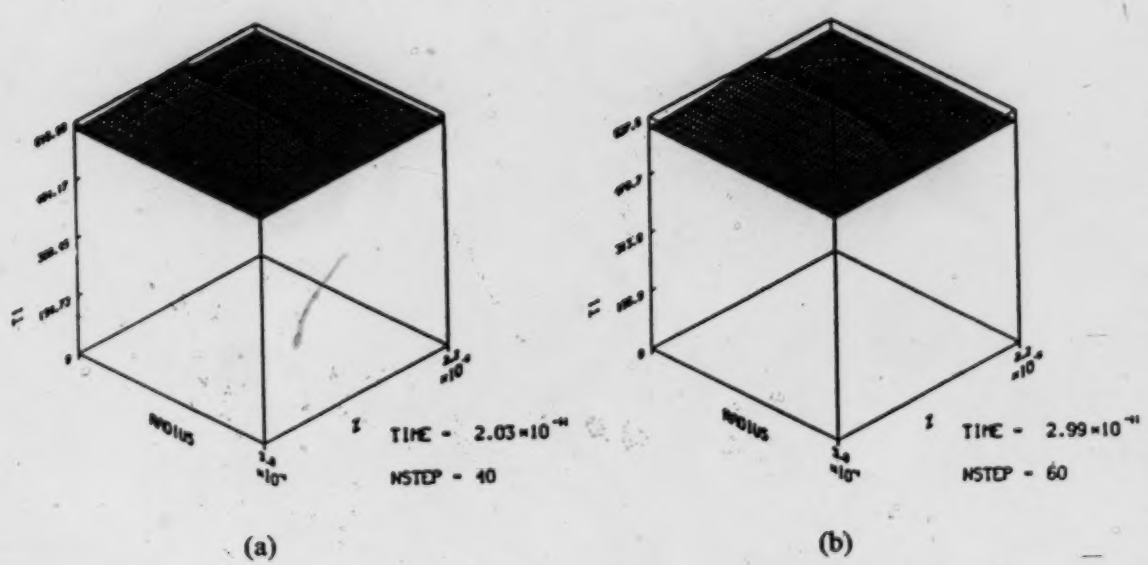


Fig. 5.3.7 Time and spatial evolution of laser intensity with PM forces are on.



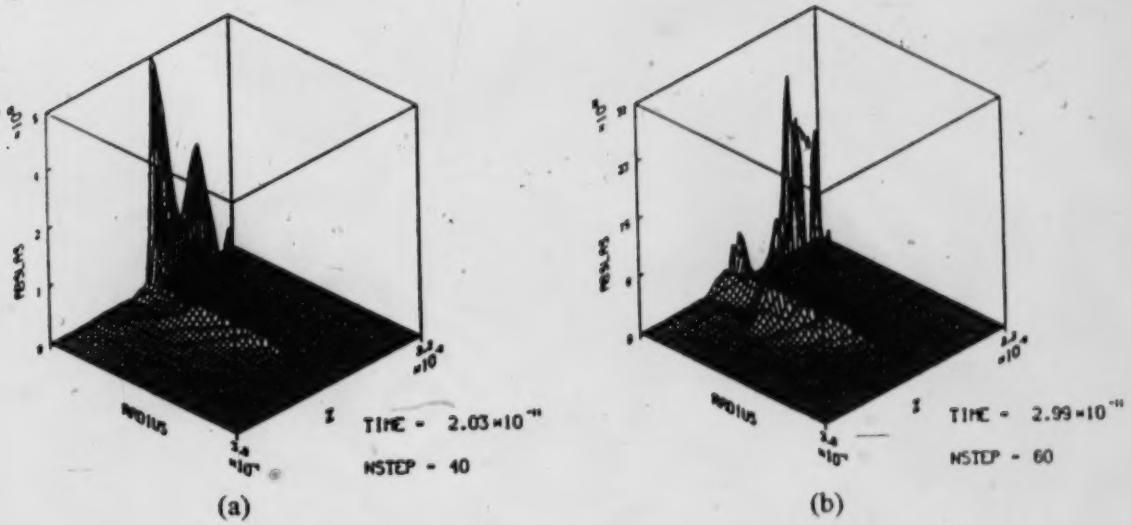
(a) (b)

Fig. 5.3.8 Electron temperature profiles with PM forces are off.



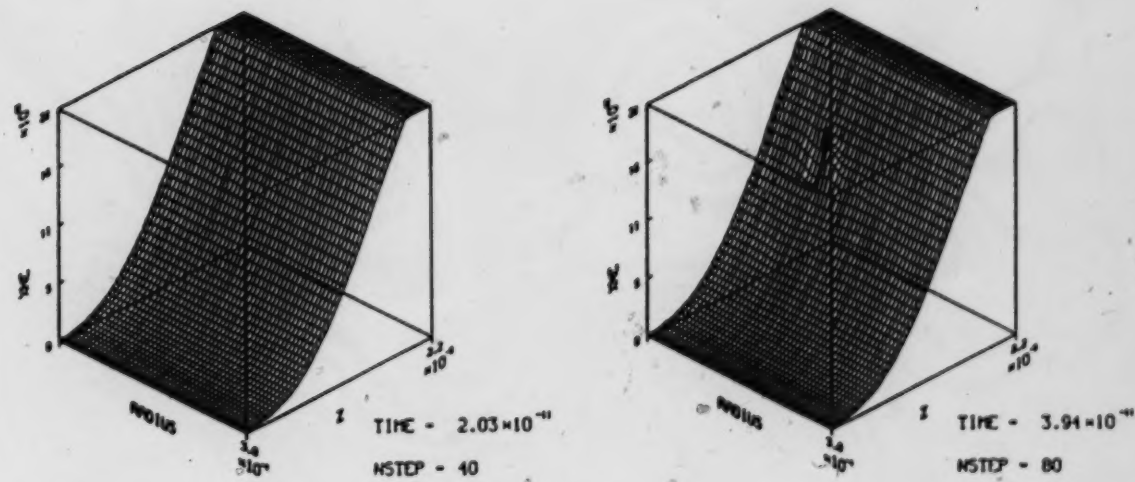
(a) (b)

Fig. 5.3.9 Ion temperature profiles with PM forces are off.



(a) (b)

Fig. 5.3.10 Laser absorption profiles with PM forces are off.



(a) (b)

Fig. 5.3.11 Electron density profiles with PM forces are off.

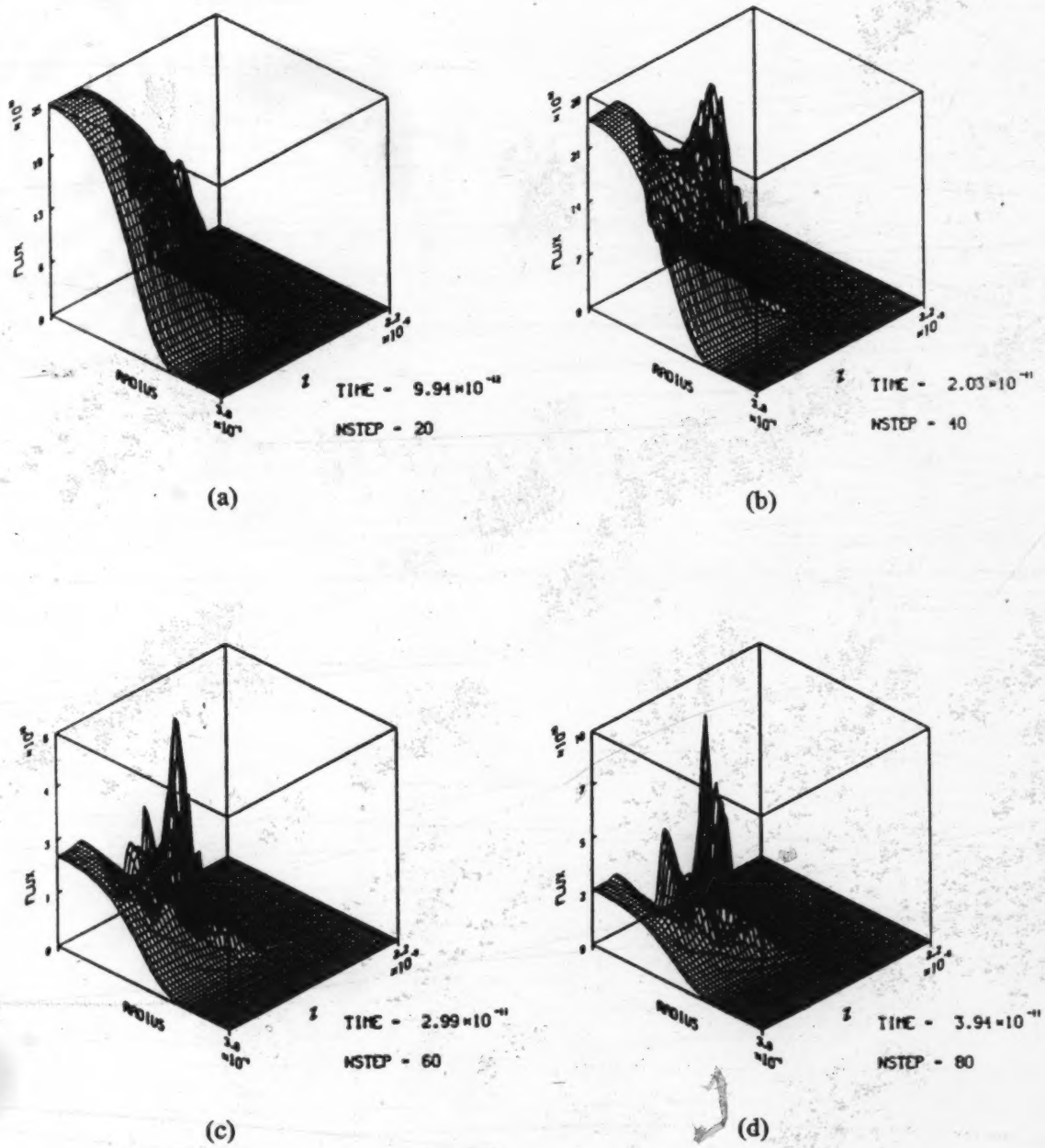


Fig. 5.3.12 Time and spatial evolution of laser intensity with PM forces are off.

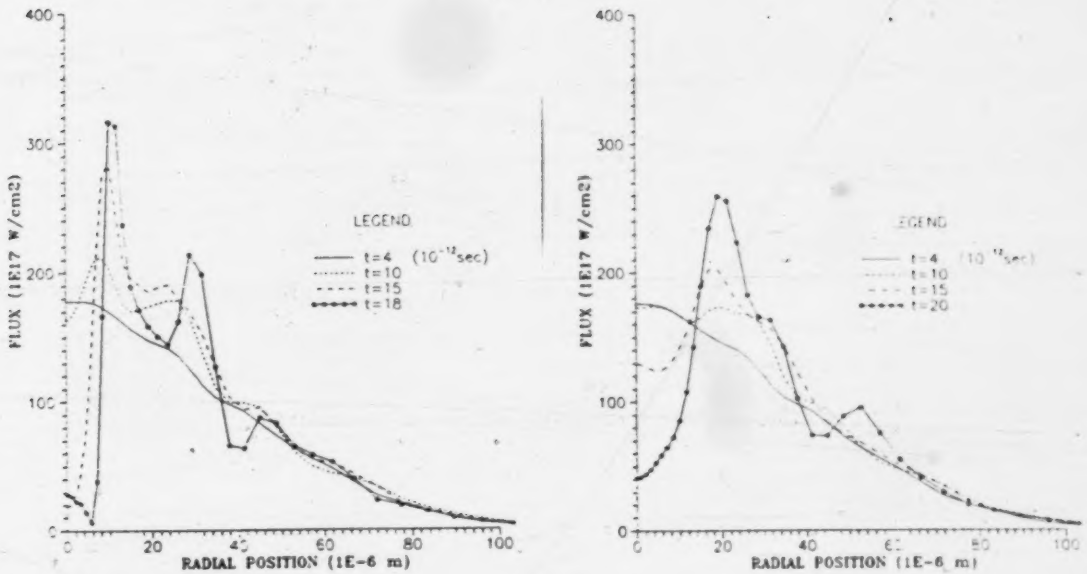


Fig. 5.4.1 Laser intensity profile at z=1.5 mm (0.5 n_c)
(a) PM on. (b) PM off.

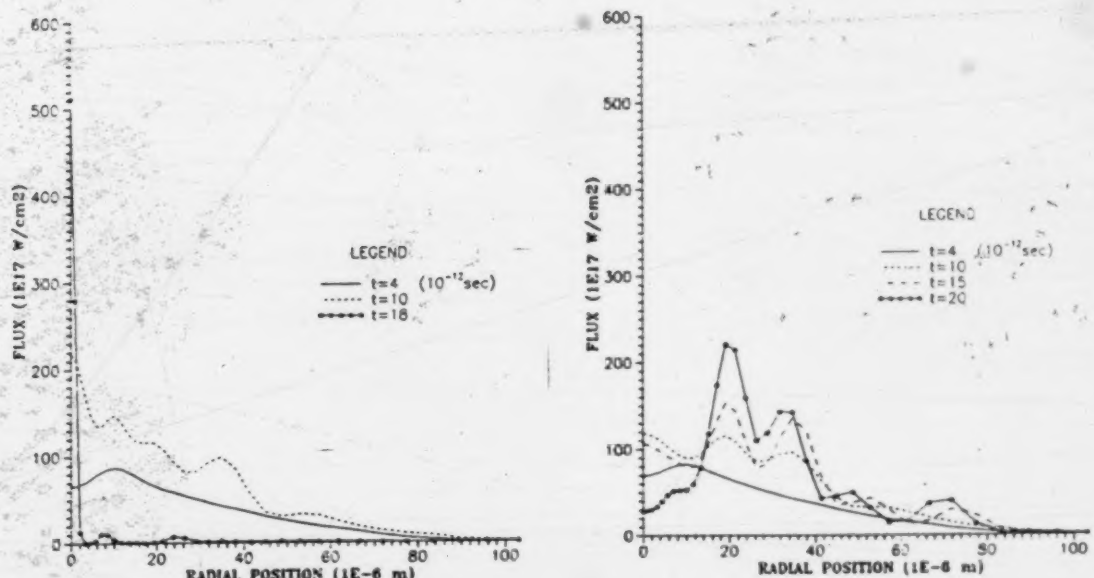


Fig. 5.4.2 Laser intensity profile at z=1.7 mm (0.75 n_c)
(a) PM on. (b) PM off.

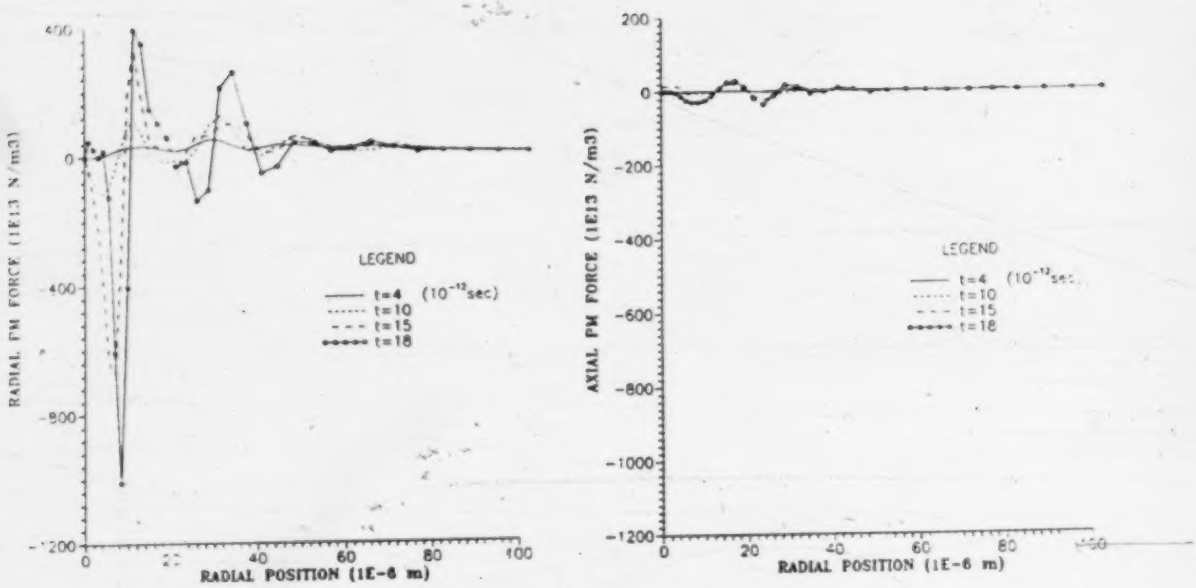


Fig. 5.4.3 Radial and axial ponderomotive force at z=1.5 mm (0.5 n_c)

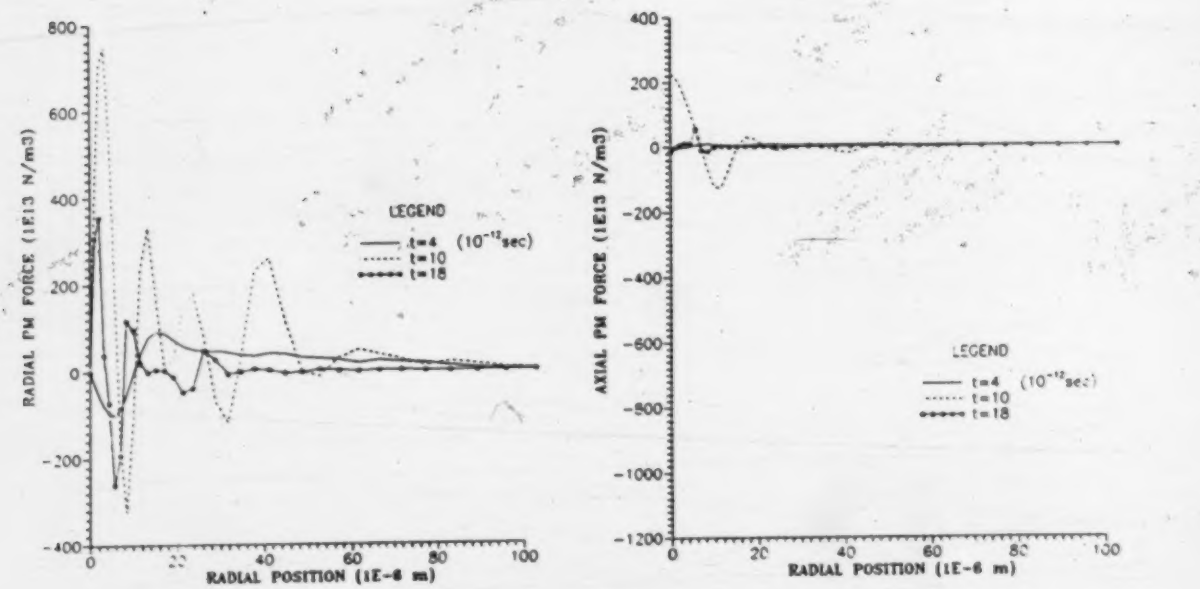


Fig. 5.4.4 Radial and axial ponderomotive force at z=1.7 mm (0.75 n_c)

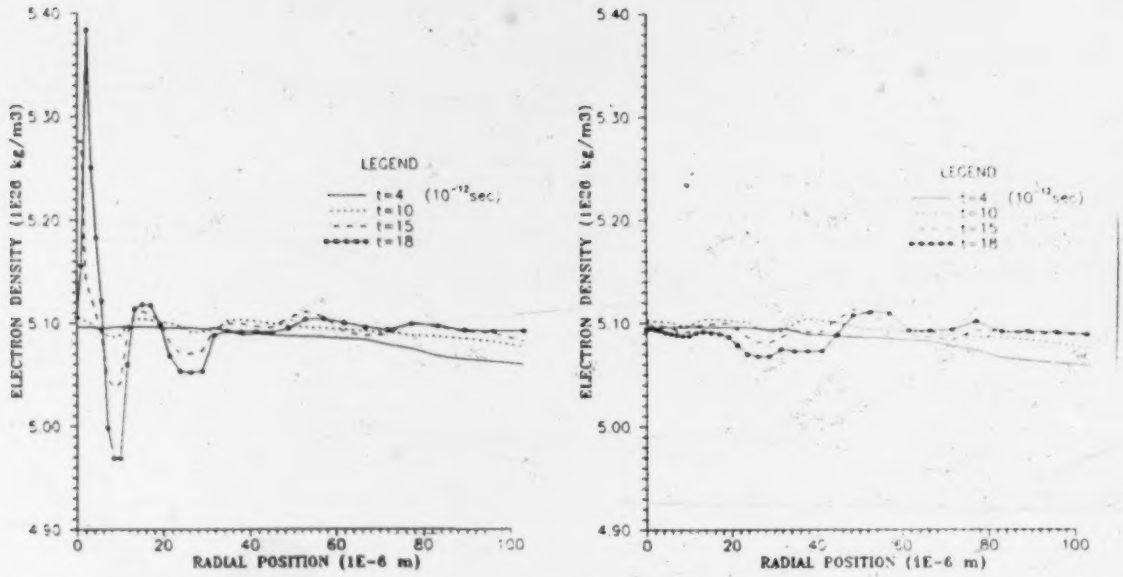


Fig. 5.4.5 Electron density profile at z=1.5 mm (0.5 n_c)
(a) PM on. *(b) PM off.*

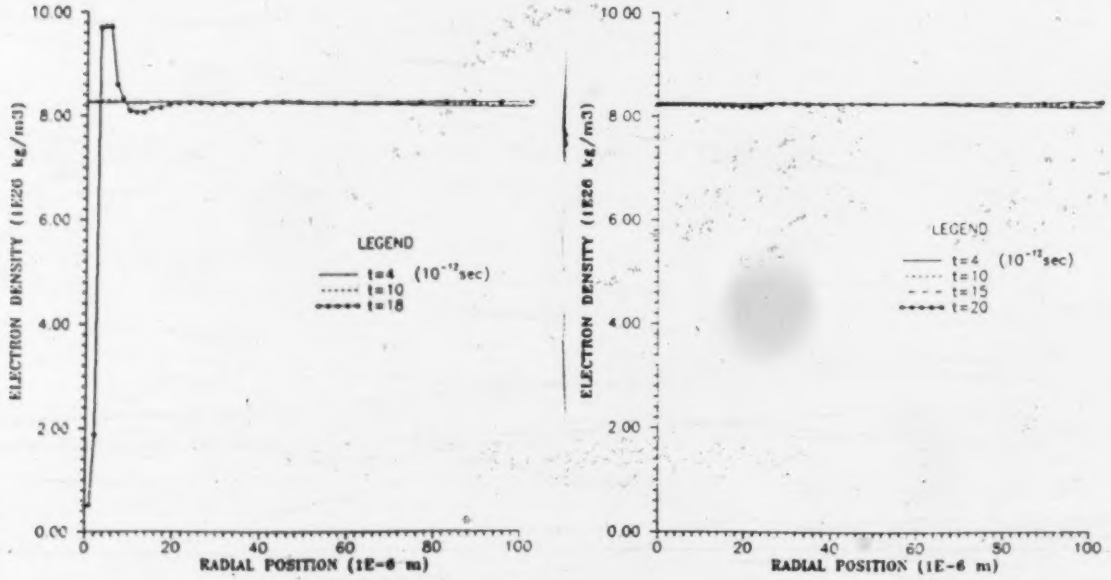


Fig. 5.4.6 Electron density profile at z=1.7 mm (0.75 n_c)
(a) PM on. *(b) PM off.*

CHAPTER 6
CONCLUSION

In this research, the HEATER routine has been modified to work best on the CRAY XMP and CYBER 205 supercomputers. The implicit vectorization method has been implemented throughout the HEATER routine to ensure portability to other types of computers and compatibility with the standard FORTRAN 77. The discussion on vectorization of a FORTRAN program has been presented in Chapter 2 followed by the description of the theory and numerical solution of a laser beam propagation model (HEATER) in Chapter 3. In Chapter 4, the HEATER's structure and how it is implemented on the CRAY XMP and CYBER 205 computers have been investigated. In Chapter 5, an example of the onset of laser light filamentation has been examined using the HEATER routine in the CASTOR hydrodynamic code to simulated the laser-plasma interaction processes.

One objective of this work is to gain speed (when running with the CRAY XMP and CYBER 205) through the vectorization of HEATER. The objective has been met by restructuring the code and by introducing new vectorized algorithms. The cyclic reduction algorithm for solving tridiagonal matrices has been used instead of the standard Gaussian elimination method. The performance increase is a factor of 3 by comparing the 'scalar mode' and 'vector mode' execution times on both the CRAY XMP and CYBER 205. One disadvantage when vectorizing HEATER is the use of implicit vectorization for which performance may not be fully exploited. However, the compatibility and portability of the

routine are preserved.

Another objective of this research is to account for the role of PM forces (radial and axial) in a self-consistent manner. Since the PM effects are not part of the analysis of the beam propagation process, they must be carefully examined and incorporated in such a way that conservation of energy must be preserved throughout the simulation. The results are satisfactory and energy conservation is maintained. The relative energy error is found to be a fraction of a percent (less than 1%).

The present research leads to improvements in computational speed; however, it also results in new questions for which further investigation is needed. In the case of vectorization, implicit vectorization (optimization mainly through DO loop structure) ensures portability and compatibility; however, it may not guarantee full performance increase. Explicit vectorization is an alternative way to increase performance; however, it is machine dependent. The main difficulty and time consuming part, (before any programmer tries to vectorize the program source code), is that the programmer must understand the vectorization inhibitors and hardware structure for each type of computers (ie. register-to-register or memory-to-memory machines). Therefore, in order to surpass these difficulties, an auto-vectorized compiler should be developed to automatically optimize the program source code when running both in scalar and vector computers. If this is to happen, then there should be a standardized FORTRAN language that is more suitable for vector computers and any other types of computers ranging from micro to supercomputers. In

routine are preserved.

Another objective of this research is to account for the role of PM forces (radial and axial) in a self-consistent manner. Since the PM effects are not part of the analysis of the beam propagation process, they must be carefully examined and incorporated in such a way that conservation of energy must be preserved throughout the simulation. The results are satisfactory and energy conservation is maintained. The relative energy error is found to be a fraction of a percent (less than 1%).

The present research leads to improvements in computational speed; however, it also results in new questions for which further investigation is needed. In the case of vectorization, implicit vectorization (optimization mainly through DO loop structure) ensures portability and compatibility; however, it may not guarantee full performance increase. Explicit vectorization is an alternative way to increase performance; however, it is machine dependent. The main difficulty and time consuming part, (before any programmer tries to vectorize the program source code), is that the programmer must understand the vectorization inhibitors and hardware structure for each type of computers (ie. register-to-register or memory-to-memory machines). Therefore, in order to surpass these difficulties, an auto-vectorized compiler should be developed to automatically optimize the program source code when running both in scalar and vector computers. If this is to happen, then there should be a standardized FORTRAN language that is more suitable for vector computers and any other types of computers ranging from micro to supercomputers. In

other words, a standard FORTRAN language should have statements, while compiling, that are independent of machine architecture.

In the MHD code CASTOR, the hydrodynamic equations are based on Eulerian coordinates from which the hydro meshes are fixed throughout the simulation processes. The problem lies when the laser beam is irradiated on a solid target and the plasma formed is allowed to expand freely. Therefore, by chosen fixed mesh, the hydrodynamic effects cannot be described correctly in the early phase of laser-plasma interaction process. In order to capture all the physical effects in an early stage, hydrodynamic meshes should be allowed to move in an Lagrangian fashion according to the rate of which plasma is expanding.

In CASTOR, there are number of different processes by which laser light heats the surrounding plasma in a laser-irradiated target. The energy is deposited primarily into electrons and the laser absorption is mainly through classical inverse bremsstrahlung. The energy transport and loses through X-ray radiation have not been accounted for in the code. This can be important effects especially when considering the study of laser-produced X-ray lasers.

In HEATER, the ponderomotive effects should somehow be included in the propagation equation and treated as a damping factor. If this can be resolved, beam propagation could be described more accurately.

The principle of laser-driven nuclear fusion and laser-produced X-ray lasers involving laser-plasma interaction processes are very complex to deal with. Perhaps, a three-dimensional beam propagation code together with three-dimensional hydrodynamic code should be

developed to describe a more complete model of the behavior of laser-produced plasmas. Some progress has been made in this regard recently at laboratories such as the Lawrence Livermore National Laboratory.

In conclusion, a radiation transport model, the effects of X-ray radiation, a moving hydro mesh (Lagrangian fashion), and a three-dimensional time dependent geometry, if included, would give such a MHD code (ie. CASTOR) a considerably more complete physical description of laser-driven plasma behavior. This would enrich the understanding of physics of laser-irradiated targets and lay a ground foundation to better understand the laser-plasma interaction processes.

BIBLIOGRAPHY

[1] Siu-Ki Stephen Au, M.Sc. thesis, University of ALberta, 1981.

[2] Pitt Lawrence, M.Sc. thesis, University of Alberta, 1977.

[3] J. Nuckolls, L. Wood, A. Thiessen, and G. Zimmerman, Nature, 239, Sept. 1972.

[4] Dawson, J.M., R.E Kedder, A. Hertzberg, G.C. Vlases, H.G. Alstrom, L.C. Steinhauer, Plasma Physics and Controlled Nuclear Fusion Research, (International Atomic Energy Agency, Vienna), Vol. I, 1971, p. 673.

[5] S Atzeni, Paper presented at Int. Meeting on Parallel Computing, Verona, Italy, 1988 (Chapter 4), p. 169.

[6] Mark H. Emery, John H. Gardner, and Jay P. Boris, Phys. fluids 27 (5), May 1984, p. 1338.

[7] M. D. Feit and J. A. Fleck, Jr., Applied Phys. Let. Vol. 29 (4), Aug. 1976, p. 234.

[8] D. J. Nicholas and S. G. Sajjadi, J. Phys D: Appl. Phys. 19, 1986, p. 737.

[9] S Sartang, R G Evans and W T Toner, J. Phys. D: Appl. Phys. 16, 1983, p. 955.

[10] M. D. Feit and J. A. Fleck, Jr., Applied Phys. Let. Vol. 28 (6), Aug. 1976, p. 331.

[11] M. D. Feit and J. A. Fleck, Jr., Applied Phys. Let. Vol. 28 (3), Feb. 1976, p. 121.

[12] W. L. Kruer and Kent Estabrook, Phys. Fluids 28, 1985, p.430.

[13] M. Hubbard and A. Montes, Preprint from Euratom/UKAEA Fusion Association, Culham Laboratory, Abingdon, Oxon., UK.

[14] Judith Harte and George Zimmerman, Preprint for presentation at the 19th Meeting of the American Phys. Soc. / Divison of Plasma Physics. Nov. 1977.

[15] R. S. Craxton and R. L. McCrory, J. Appl. Phys. 56 (1), 1984, p.108.

[16] Frederick Tappert, J. Opt. Soc. Am. 66 (12) 1976, p.1368.

[17] R. D. Jones et al., Phys. Fluids 31 (5), 1988, p. 1249.

[18] H. D. Dudder, Comp. Phys. Comm. 10, 1975, p. 155.

[19] D. Havazelet, C. E. Capjack, D. Shwarts and R. Marchand, Phys. Fluids B 1 (4), 1989, p. 893.

[20] R. Marchand, R. Rankin, C. E. Capjack, and A. Birnboim, Phys. Fluids 30 (5), 1987, p. 1521.

[21] R. Rankin, C. E. Capjack, and C. R. James, Phys. Rev. Lett. 63 (15), 1989, p. 1597.

[22] R. Rankin, R. Marchand, and C. E. Capjack, Phys. Fluids 31 (8), 1988, p. 2327.

[23] R Marchand, C. E. Capjack, and C. R. James, J. Appl. Phys. 57 (3), 1985, p. 732.

[24] J. N. McMullin, C. E. Capjack, and C. R. James, Phys. Fluids 21 (10), 1978, p. 1828.

[25] D. Giulietti, A. Giulietti, M. Lucchesi and M. Vaselli, J. Appl. Phys. 58 (8) 1985, p. 2916.

[26] P. K. Tien, J. P. Gordon, and J. R. Whinnery, Proc. of the IEEE. p. 129.

[27] F. P. Mattar, Comp. Phys. Comm. 22, 1981, p. 1.

[28] Steinhauer, L. C. and H. G. Ahlstrom, Phys. of Fluids., 14, 1971, p.81.

[29] Suckewer, S., C. H. Skinner, D. R. Voorhees, H. M. Milchberg, C. Keane and A. Semet, IEEE J. Quan. Elect., QE-19, 1983, p.1855.

[30] G.L.Pert, Plasma Physics and Nuclear Fusion Research, Academic Press, 1981, Chap. 25.

[31] C. J. Keane, N. M. Ceglio, B. J. Macgowan, D. L. Matthews, D. G. Nilson, J. E. Trebes and D. A. Whelan, J. Phys. B: At. Mol Opt. Phys., 22, 1989, p. 3343.

[32] Richard A London, Mordecai D Rosen, M Stephen Maxon, David C Eder and Peter L Hagelstein, J. Phys. B: At. Mol. Opt. Phys., 22, 1989, p. 3363.363.

[33] C. L. S. Lewis et al., Plasma Physics and Controlled Fusion, 30 (1), 1988, p. 35.

[34] Dennis L. Matthews and Mordecai D. Rosen, Scientific American Dec. 1988, p. 86.

[35] Canahan, Applied Numerical Method, Prentice Hall Inc., 1977, Chapter 7.

[36] J. M. Buchlin, Vector Computers And Computational Fluid Dynamics, Von Karman Institute for Fluid Dynamics, 1986.

[37] John M. Levesque & Joel W. Williamson, A Guide Book To FORTRAN On Supercomputers, Academic Press Inc., 1989.

[38] C. A. Brebbia & A. Peters, Applications Of Supercomputers In Engineering: Algorithms, Computer Systems And User Experience, Proc. of First Int. Conf., Sep. 1989.

[39] R. W. Hockney & C. R. Jesshope, Parallel Computer Architecture: Programming And Algorithms,1981.

[40] J. F. Janni, Large FORTRAN Program Structure, Maintenance, Reliability, And Documentation, Fifth Annual Review of Progress in App. Comp. Electromagnetic, March 1989.

[41] A. M. Davies and R. Proctor, Computer & Fluids, Vol. 18 No. 3 pp. 259-270, 1990.

[42] S. Atzeni, Meeting on Parallel Computing, Chap. 4, 1988.

[43] Hui Cheng, Comp. & Fluid, 1989.

[44] Cliff Morgan, The U of A Computing Services Bulletin, May 1986.

[45] CRAY Computer Systems CFT77 Reference Manual, CRAY Research Inc., 1986.

[46] FORTRAN 200 Version 1 Reference Manual, Control Data Corp., 1987.

[47] David Kershaw, Parallel Computations, Academic Press. Inc., 1882.

[48] J. N. McMullin, C. E. Capjack and C. R. James, Comp. Phys. Comm. 23, 1981, p. 31.

[49] J. A. Fleck, Jr., J. Comput. Phys. 16, 1974, p. 324.

[50] J. P. Christiansen and N. K. Winsor, Comp. Phys. Comm. 17, 1979 p. 397.

[51] William L. Kruer, The Physics of Laser Plasma Interaction. Copyright 1988 by Addison-Wesley Publishing Company Inc.

[52] Heinrich Hora, Physics of Laser Driven Plasma, Copyright 1981 by Hohn Wiley & Sons, Inc.

[53] Johnston and Dawson, The Physics of Fluids, 16(3), 1973, p. 722.

[54] Chen, Introduction to Plasma Physis and Control Nuclear Fusion, Prentice Hall, 1984.

[55] A. Bruce Langdon, Phys. Rev. Let., 44 (9), 1980, p. 575.

[56] Joseph T. Verdeyen, Laser Electronic, Chap. 3, Prentice-Hall Inc. 1981.

[57] J. P. Christiansen and N. K. Windsor, J. of Comp. Phys. 35 (3), 1980 p. 291.

[58] J. P. Christiansen and K. V. Roberts, Comp. Phys. Comm., 7, 1974, p. 245.

[59] J. P. Christiansen, D.E.T.F. Ashby, and K. V. Roberts, Comp. Phys. Comm., 7, 1974, p. 271.

APPENDIX 1

DERIVATION OF WAVE EQUATION

The wave equation is derived by starting with the classical Maxwell's equations:

$$\nabla \mathbf{x} \mathbf{E} = -\frac{1}{c}\frac{\partial \mathbf{B}}{\partial t} = -\frac{1}{c}\frac{\partial \mathbf{H}}{\partial t} \qquad \text{(A1.1)}$$

$$\nabla \mathbf{x} \mathbf{H} = \frac{4\pi}{c}\mathbf{J} + \frac{1}{c}\frac{\partial \mathbf{D}}{\partial t} = \frac{4\pi}{c}\mathbf{J} + \frac{1}{c}\frac{\partial \mathbf{E}}{\partial t} \qquad \text{(A1.2)}$$

$$\nabla . \mathbf{D} = 4\pi\rho \qquad \text{(A1.3)}$$

The generalized Ohm's law is given by[1]:

$$\frac{m}{e^2 n_e}\left(\frac{d\mathbf{J}}{dt} + \upsilon \mathbf{J}\right) = \mathbf{E} + \frac{1}{c}\,\mathbf{v}\mathbf{x}\mathbf{H} + \frac{1}{en_e c}\,\mathbf{J}\mathbf{x}\mathbf{H} + \frac{c}{en_e}\,\frac{\nabla p}{1 + 1/Z}$$

If the Lorentzian term $\mathbf{v}\mathbf{x}\mathbf{H}$, the Hall term $\mathbf{J}\mathbf{x}\mathbf{H}$, and the electron pressure ∇p are neglected, then the above equation becomes:

$$\frac{d\mathbf{J}}{dt} + \upsilon \mathbf{J} = \frac{\omega_p^{\,2}}{4\pi}\mathbf{E} \qquad \text{(A1.4)}$$

where υ is the electron collision frequency and ω_p is the plasma frequency.

If the quantities **E**, **H**, **J** are of periodic time dependence with a frequency ω, then **E**, **H**, and **J** can be expressed as:

$$\mathbf{E} = \mathbf{E}_r \exp(i\omega t) \qquad \text{(A1.5)}$$

$$\mathbf{H} = \mathbf{H}_r \exp(i\omega t) \qquad \text{(A1.6)}$$

$$\mathbf{J} = \mathbf{J}_r \exp(i\omega t) \qquad \text{(A1.7)}$$

where $\mathbf{E}_r$, $\mathbf{H}_r$ and $\mathbf{J}_r$ depend only on the spatial coordinate, then the time independent Maxwellian equation becomes:

1 Henrich Hora, Theory of Laser-Plasma Interaction, Prentice Hall, 1980.

$$\nabla x E_r = - \frac{i\omega}{c} H_r \qquad \text{(A1.8)}$$

$$\nabla x H_r = \frac{4\pi}{c} J_r + \frac{1}{c} i\omega E_r \qquad \text{(A1.9)}$$

An expression for conductivity σ may be obtained from the simplified Ohm's law. This is given by:

$$i\omega J_r + \upsilon J_r = \frac{\omega_p^2}{4\pi} E_r$$

$$J = \frac{\omega_p^2}{4\pi i\omega \left[1 - \frac{i\upsilon}{\omega}\right]} E_r = \sigma E_r \qquad \text{(A1.10)}$$

The substitution of Eqs. A1.10 into A1.9 becomes:

$$\nabla x H_r = - \frac{i\omega_p^2}{c\omega \left[1 - \frac{i\upsilon}{\omega}\right]} E_r + \frac{i\omega}{c} E_r. \qquad \text{(A1.11)}$$

By taking the curl of Eq. A1.8 and using the vector identity

$$\nabla x \nabla x E = \nabla(\nabla . E) - \nabla^2 E \qquad \text{(A1.12)}$$

gives

$$\nabla x(\nabla x E_r) = - \frac{i\omega}{c} (\nabla x H_r) \qquad \text{(A1.13)}$$

$$\text{or} \quad \nabla x H_r = \frac{ic}{\omega} \left[\nabla(\nabla . E_r) - \nabla^2 E_r\right] \qquad \text{(A1.14)}$$

The substitution of Eq. A1.11 into the above expressions yields:

$$\frac{ic}{\omega} \left[\nabla(\nabla . E_r) - \nabla^2 E_r\right] = - \frac{i\omega_p^2}{c\omega \left[1 - \frac{i\upsilon}{\omega}\right]} E_r + \frac{i\omega}{c} E_r \qquad \text{(A1.15)}$$

The above equation may be rearranged to give:

$$\nabla^2 E + \frac{\omega^2}{c^2} \varepsilon E - \nabla(\nabla . E) = 0 \qquad \text{(A1.16)}$$

$$\text{where } \varepsilon = 1 - \frac{\omega_p^2}{\omega^2 \left[1 - \frac{i\upsilon}{\omega}\right]} = \left[1 - \frac{\omega_p^2}{\omega^2 + \upsilon^2}\right] + i \frac{\omega_p^2 \upsilon \omega}{\omega^2(\omega^2 + \upsilon^2)} \qquad \text{(A1.17)}$$

ε is a complex dielectric constant. For an electromagnetic wave, the propagation vector **k** is perpendicular to the electric field **E**; therefore,

$$\nabla.\mathbf{E} = i\mathbf{k}.\mathbf{E} = 0. \qquad \text{(A1.18)}$$

The subscript r is drop from the above expression since **E** depends only on spatial coordinate, the term $\nabla.\mathbf{E}$ is neglected, then Eq. A1.16 becomes:

$$\nabla^2\mathbf{E} + \frac{\omega^2}{c^2}\varepsilon\mathbf{E} = 0 \qquad \text{(A1.19)}$$

In the colisionless or weakly ionized plasma (ie. underdense region) collision frequency is much less than the laser frequency, then the term $\omega^2 + \upsilon^2$ is approximated to ω^2; Therefore, expression A1.17 reduces to:

$$\varepsilon = 1 - \frac{\omega_p^2}{\omega^2}\left[1 - i\,\frac{\upsilon}{\omega}\right] \qquad \text{(A1.20)}$$

APPENDIX 2

DERIVATION OF PONDEROMOTIVE FORCES

The ponderomotive forces are derived according to Chen's analysis[1]. As a laser beam is focused onto a plasma medium, the intensity at the focal region is high. The electromagnetic field which is proportional to intensity rises to a very large value. Therefore, the field imposes a strong Lorentz force which is expressed in the form of electron equation of motion as:

$$m_e \frac{d\mathbf{v}}{dt} = -e \left[\mathbf{E}(\mathbf{r}) + \frac{1}{c} \mathbf{v} \times \mathbf{B}(\mathbf{r}) \right] \qquad \text{(A2.1)}$$

where m_e is the mass of electron, $\mathbf{v}$ is the velocity of electron, $\mathbf{E}(\mathbf{r})$ and $\mathbf{B}(\mathbf{r})$ are the electric field and magnetic field of the wave respectively.

Assuming an incident electric field is in the form:

$$\mathbf{E}(r,t) = \mathbf{E}_s(r)\cos(\omega t) \qquad \text{(A2.2)}$$

where ω is the angular frequency of the incident wave, $\mathbf{E}_s(r)$ is the electric field with spatial dependence only. To the first order, the force on an electron at location $\mathbf{r}$ can be evaluated by taking $\mathbf{E}$ at the initial position. Neglecting the term $\mathbf{v} \times \mathbf{B}$ since it is smaller than $\mathbf{E}$ by a factor of v/c, the force is evaluated at the initial position as:

$$m_e \frac{d\mathbf{v}^1}{dt} = -e \left[\mathbf{E}(r_0) \right] \qquad \text{(A2.3)}$$

Integrating the above expression using the condition that the electron has zero velocity at the maximum amplitude of oscillation,

---

[1] Francis F. Chen, Introduction to Plasma Physics and Control Fusion, Plenum Press Inc., 1984.

then the first order velocity is:

$$v^1 = -\frac{e}{m_e \omega} E_s(r)\sin(\omega t) = \frac{dr_1}{dt} \qquad \text{(A2.4)}$$

The first order displacement perturbation of the electron from the initial position is calculated by integration of Eq. A2.4 yielding:

$$\delta r^1 = \int v^1 dt = \frac{e}{m_e \omega^2} E_s(r)\cos(\omega t) \qquad \text{(A2.5)}$$

Using the Equation A2.1, the second order approximation of the force about the initial position is found to be:

$$m_e \frac{dv^2}{dt} = -e\left[(\delta r^1 . \nabla)E(r) + \frac{1}{c} v^1 \times B(r)\right]_{r=r0} \qquad \text{(A2.6)}$$

where v^2 is the second order correction term to the velocity v.

From the Maxwell's equation

$$\frac{1}{c}\frac{\partial B}{\partial t} = -\nabla \times E \qquad \text{(A2.7)}$$

Using Eq. A2.2, the term B from the above expression can be express as:

$$B(r) = \frac{c}{\omega} \nabla \times E_s(r)\sin(\omega t) \qquad \text{(A2.8)}$$

Substituting Eqs. A2.8 into A2.6 and averaging over time (using $\langle \sin^2\omega t \rangle = \langle \cos^2\omega t \rangle = \frac{1}{2}$) gives:

$$m_e \left\langle \frac{dv^2}{dt} \right\rangle = -\frac{e^2}{m_e \omega^2} \frac{1}{2}\left[(E_s . \nabla)E_s + E_s \times \nabla \times E_s\right] \qquad \text{(A2.9)}$$

From the vector identity:

$$\frac{1}{2}\nabla E_s^2 = (E_s . \nabla)E_s + E_s \times \nabla \times E_s$$

The Equation (A2.9) can be written as:

$$F_{NL} = m_e \left\langle \frac{dv^2}{dt} \right\rangle = -\frac{1}{4} \frac{e^2}{m_e \omega^2} \nabla E_s^2 \qquad \text{(A2.10)}$$

Using the fact that $E_s^2 = 2\langle E^2 \rangle$ and $\omega_p^2 = \frac{4\pi ne^2}{m_e}$ the above equation is written as:

$$F_{NL} = -\frac{\omega_p^2}{\omega^2} \frac{\nabla \langle E^2 \rangle}{8\pi} \qquad \text{(A2.11)}$$

An alternative approach in obtaining the ponderomotive force equation is to start from the fluid equation. Neglecting the pressure term, the fluid equation can be written as follows:

$$m_e \left[\frac{\partial \mathbf{u}_e}{\partial t} + (\mathbf{u}_e \cdot \nabla)\mathbf{u}_e \right] = -e \left[\mathbf{E} + \frac{1}{c} (\mathbf{u}_e \times \mathbf{B}) \right] \qquad \text{(A2.12)}$$

where m_e = mass of electron,

e = electron charge,

$\mathbf{u}_e$ = electron velocity.

Let the response of a homogeneous plasma to a high frequency field whose amplitude is spatial dependent.

$$\mathbf{E} = \mathbf{E}_s \sin(\omega t) \qquad \text{(A2.13)}$$

For the first order term of $|\mathbf{E}|$; $\mathbf{u}_e = \mathbf{u}^f$, then Eq. A2.12 becomes

$$m_e \frac{\partial \mathbf{u}^f}{\partial t} = -e\mathbf{E}_s \sin(\omega t)$$

or
$$\mathbf{u}^f = \frac{e}{m\omega} \mathbf{E}_s \cos(\omega t) \qquad \text{(A2.14)}$$

Also, from Maxwell's equation

$$\frac{1}{c} \frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E} = -\nabla \times \mathbf{E}_s \sin(\omega t)$$

or
$$\mathbf{B} = \frac{c}{\omega} \nabla \times \mathbf{E}_s \cos(\omega t) \qquad \text{(A2.15)}$$

For the second order term, the fluid equation is written as:

$$m_e\left[\frac{\partial u_e^s}{\partial t} + (u_e^f.\nabla)u_e^f\right] = -e\left[E^s + \frac{1}{c}(u_e^f x B)\right] \qquad \text{(A2.16)}$$

The electrons are simply oscillating in the local electric field. Substitution of Eq. A2.14 & A2.15 into the above equation and averaging the force over these rapid oscillation yields

$$m_e\frac{\partial u_e^s}{\partial t} + m_e < u_e^f.\nabla)u_e^f >_t = -eE^s - \frac{e}{c} < u_e^f x B >_t \qquad \text{(A2.17)}$$

Using the fact that $<\sin^2\omega t> = <\cos^2\omega t> = \frac{1}{2}$, then

$$< u_e^f.\nabla)u_e^f >_t = \frac{e^2}{2m_e^2\omega^2}(E_s.\nabla)E_s \qquad \text{(A2.18)}$$

and

$$< u_e^f x B >_t = \frac{ec}{2m_e\omega^2}(E_s x \nabla x E_s) \qquad \text{(A2.19)}$$

From Eq. A2.18 & A2.19, the expression A2.17 becomes:

$$m_e\frac{\partial u_e^s}{\partial t} = -eE^s - \frac{e^2}{2m_e\omega^2}\left\{(E_s.\nabla)E_s - (E_s x \nabla x E_s)\right\} \qquad \text{(A2.20)}$$

From vector identity, the above equation can be written as:

$$m_e\frac{\partial u_e^s}{\partial t} = -eE^s - \frac{e^2}{4m_e\omega^2}\nabla E_s^2 \qquad \text{(A2.21)}$$

where the first term on the right-hand side is the self-consistent electric field and the second term refers to the ponderomotive force. It is also consistent to Eq. A2.11 derived from Chen.

APPENDIX 3

CYCLIC REDUCTION ALGORITHM

The tridiagonal is usually written in the form

$$L_i X_{i-1} + D_i X_i + U_i X_{i+1} = Wi, i = 1,2,...,N. \quad \text{(A3.1)}$$

where L_i is the lower element, D_i is the diagonal element and U_i is the upper element of the tridiagonal matrix.

1. OPERATION

The fundamental process of Cyclic Reduction is described as follows. If the index of Eq. A3.1 is shifted by (2i-1) then it becomes:

$$L_{2i-1}X_{2i-2} + D_{2i-1}X_{2i-1} + U_{2i-1}X_{2i} = W_{2i-1} \quad \text{(A3.2)}$$

If the index of Eq. A3.1 is shifted by (2i+1), it becomes:

$$L_{2i+1}X_{2i} + D_{2i+1}X_{2i+1} + U_{2i+1}X_{2i+2} = W_{2i+1} \quad \text{(A3.3)}$$

The index of Eq. A3.1 is shifted by (2i), it becomes:

$$L_{2i}X_{2i-1} + D_{2i}X_{2i} + U_{2i}X_{2i+1} = W_{2i} \quad \text{(A3.4)}$$

Multiplying Eq. A3.2 by $-L_{2i}D_{2i-1}^{-1}$ yields:

$$L_{2i-1}X_{2i-1}(-L_{2i}D_{2i-1}^{-1}) + D_{2i-1}X_{2i+1}(-L_{2i}D_{2i-1}^{-1}) + U_{2i-1}X_{2i}(-L_{2i}D_{2i-1}^{-1}) = W_{2i-1}(-L_{2i}D_{2i-1}^{-1}) \quad \text{(A3.5)}$$

Multiplying Eq. A3.3 by $-U_{2i}D_{2i+1}^{-1}$ yields:

$$L_{2i+1}X_{2i}(-U_{2i}D_{2i+1}^{-1}) + D_{2i+1}X_{2i+1}(-U_{2i}D_{2i+1}^{-1}) + U_{2i+1}X_{2i+2}(-U_{2i}D_{2i+1}^{-1}) = W_{2i+1}(-U_{2i}D_{2i+1}^{-1}) \quad \text{(A3.6)}$$

By adding Eqs. A3.4, A3.5 and A3.6, the result is

$$(-L_{2i}D^{-1}_{2i-1}L_{2i-1})X_{2i-2} + (D_{2i} - L_{2i}D^{-1}_{2i}U_{2i-1} - U_{2i}D^{-1}_{2i+1}L_{2i+1})X_{2i}$$

$$+ (U_{2i}D^{-1}_{2i+1}U_{2i+1})X_{2i+2} = W_{2i} - L_{2i}D^{-1}_{2i-1}W_{2i+1} - U_{2i}D^{-1}_{2i+1}W_{2i+1} \quad (A3.7)$$

for $i = 1,2,3,\ldots,2^p-1$. Here p is the highest power of 2 with $2^p \leq N$.

This new Equation A3.7 is a block tridiagonal system which contains only even terms. This is called the reduction step. Once X_{2i-2}, X_{2i} and X_{2i+2} are computed, these results are substituted into Equation A3.2 to compute the odd term X_{2i-1}. This is back substitution step.

2. ALGORITHM

Let $L^0_i = L_i$, $U^0_i = U_i$, $D^0_i = D_i$ and $W^0_i = W_i$ for $i = 1,2,\ldots,(N-1),N$.

Then for each $q = 0,1,2,3,\ldots,p$ we take

$$DD^q_{2i-1} = 1/D^q_{2i-1} \qquad 1 \leq 2i-1 \leq r(q,n).$$

$$LL^q_{2i-1} = L^q_{2i-1}DD^q_{2i-1} \qquad 1 \leq 2i-1 < r(q,n).$$

$$LL^q_{2i} = L^q_{2i} \qquad 1 \leq 2i < r(q,n).$$

$$UU^q_{2i-1} = U^q_{2i}DD^q_{2i+1} \qquad 1 \leq 2i-1 < r(q,n).$$

$$D^{q+1}_i = D^q_{2i} - LL^q_{2i-1}UU^q_{2i-1} - LL^q_{2i}UU^q_{2i} \qquad 1 \leq i \leq r(q+1,n).$$

$$L^{q+1}_i = - LL^q_{2i}LL^q_{2i+1} \qquad 1 \leq i < r(q+1,n).$$

$$U^{q+1}_i = - UU^q_{2i}UU^q_{2i+1} \qquad 1 \leq i < r(q+1,n).$$

$$W^{q+1}_i = W^q_{2i} - LL^q_{2i-1}W^q_{2i-1} - UU^q_{2i}W^q_{2i+1} \qquad 1 \leq i < r(q+1,n).$$

where r(q,n) is the largest integer r such that $r.2^q \leq N$.

For the solution, let

$X_1^p = DD_1^p W_1^p$ for each q = (p-1),(p-2),...,3,2,1,0 we take

$$X_{2i}^q = X_i^{q+1} \qquad 2 \leq 2i \leq r(q,n)$$

and

$$X_{2i-1}^q = DD_{2i-1}^q [W_{2i-1}^q - LL_{2i-2}^q X_{2i-2}^q - UU_{2i-1}^q X_{2i}^q]$$

$$1 \leq 2i-1 \leq r(q,n).$$

Finally, $X_i = X_i^0$ for i=1,2,...,N.

Full details on the algorithm for cyclic reduction can be found in a paper by David Kershaw[1].

1 David Kershaw, Parallel Computation, Academic Press Inc., 1982.

APPENDIX 4

DERIVATION OF CYLINDRICAL CUBIC SPLINE

From Fleck[1], if the Laplacian is assumed to be continuous and piece-wise linear, then it is expressed as:

$$\frac{1}{r}\frac{\partial}{\partial r}\left[r\,\frac{\partial \mathscr{E}}{\partial r}\right] = \frac{M_{j-1}}{l_j}(r_j - r) + \frac{M_j}{l_j}(r - r_{j-1}) \qquad (A4.1)$$

or

$$\frac{\partial}{\partial r}\left[r\,\frac{\partial \mathscr{E}}{\partial r}\right] = \frac{M_{j-1}}{l_j}(rr_j - r^2) + \frac{M_j}{l_j}(r^2 - rr_{j-1}) \qquad (A4.2)$$

for $r_{j-1} < r < r_j$ and $j = 1,2,3,\ldots,j_m$ and $l_j = r_j - r_{j-1}$.

The integration of Eq. A4.2 gives:

$$\int\frac{\partial}{\partial r}\left[r\,\frac{\partial \mathscr{E}}{\partial r}\right]dr = \int\left[\frac{M_{j-1}}{l_j}\left(rr_j - r^2\right) + \frac{M_j}{l_j}\left(r^2 - rr_{j-1}\right)\right]dr \qquad (A4.3)$$

$$\frac{\partial \mathscr{E}}{\partial r} = \frac{M_{j-1}}{l_j}\,r\left[\frac{r_j}{2} - \frac{r}{3}\right] + \frac{M_j}{l_j}\,r\left[\frac{r}{3} - \frac{r_{j-1}}{2}\right] + \frac{C_j}{r} \qquad (A4.4)$$

The integration of Eq. A4.4 with respect to r gives:

$$\mathscr{E} = \frac{M_{j-1}}{l_j}\,r^2\left[\frac{r_j}{4} - \frac{r}{9}\right] + \frac{M_j}{l_j}\,r^2\left[\frac{r}{9} - \frac{r_{j-1}}{4}\right] + C_j\ln r + D_j \qquad (A4.5)$$

In the boundary condition on the electric field is applied at the origin, it requires that:

$$C_1 = 0, \text{ and } D_1 = \mathscr{E}_0 \qquad (A4.6)$$

and

$$\mathscr{E}(r_j) = \mathscr{E}_j \text{ for } j = 0,1,2,\ldots,j_m \qquad (A4.7)$$

Let investigate at $r = r_j$. Substituting Eq. A4.7 into A4.5 then

$$\mathscr{E}_j = \frac{M_{j-1}}{l_j}\,r_j^2\left[\frac{r_j}{4} - \frac{r_j}{9}\right] + \frac{M_j}{l_j}\,r_j^2\left[\frac{r_j}{9} - \frac{r_{j-1}}{4}\right] + C_j\ln r_j + D_j \qquad (A4.8)$$

1

J. A. Fleck, Jr., J. Comp. Phys., 16, 1974, p. 324-341.

At $r = r_{j-1}$

$$\mathscr{E}_{j-1} = \frac{M_{j-1}}{I_j} r_{j-1}^2\left[\frac{r_j}{4} - \frac{r_{j-1}}{9}\right] + \frac{M_j}{I_j} r_{j-1}^2\left[\frac{r_{j-1}}{9} - \frac{r_{j-1}}{4}\right] + C_j \ln r_{j-1} + D_j \quad \text{(A4.9)}$$

To solve for C_j, subtracting Eqs. A4.9 from A4.8 yields

$$\mathscr{E}_j - \mathscr{E}_{j-1} = \frac{M_{j-1}}{I_j} r_j^2\left[\frac{r_j}{4} - \frac{r_j}{9}\right] + \frac{M_j}{I_j} r_j^2\left[\frac{r_j}{9} - \frac{r_{j-1}}{4}\right] - \frac{M_{j-1}}{I_j} r_{j-1}^2\left[\frac{r_j}{4} - \frac{r_{j-1}}{9}\right] - \frac{M_j}{I_j} r_{j-1}^2\left[\frac{r_{j-1}}{9} - \frac{r_{j-1}}{4}\right] + C_j(\ln r_j - \ln r_{j-1}) \quad \text{(A4.10)}$$

$$C_j\left[\ln\frac{r_j}{r_{j-1}}\right] = (\mathscr{E}_j - \mathscr{E}_{j-1}) + \frac{M_{j-1}}{I_j}\left[r_{j-1}^2\left[\frac{r_j}{4} - \frac{r_{j-1}}{9}\right] - r_j^2\left[\frac{r_j}{4} - \frac{r_j}{9}\right]\right] + \frac{M_j}{I_j}\left[r_{j-1}^2\left[\frac{r_{j-1}}{9} - \frac{r_{j-1}}{4}\right] - r_j^2\left[\frac{r_j}{9} - \frac{r_{j-1}}{4}\right]\right] \quad \text{(A4.11)}$$

Therefore:

$$C_j = \left\{(\mathscr{E}_j - \mathscr{E}_{j-1}) + \frac{M_{j-1}}{I_j}\left[r_{j-1}^2\left[\frac{r_j}{4} - \frac{r_{j-1}}{9}\right] - r_j^2\left[\frac{r_j}{4} - \frac{r_j}{9}\right]\right] + \frac{M_j}{I_j}\left[r_{j-1}^2\left[\frac{r_{j-1}}{9} - \frac{r_{j-1}}{4}\right] - r_j^2\left[\frac{r_j}{9} - \frac{r_{j-1}}{4}\right]\right]\right\} / \left[\ln\frac{r_j}{r_{j-1}}\right] \quad \text{(A4.12)}$$

Similarly, in order to find D_j, Eq. A4.8 is multiplied by $\ln r_{j-1}$ and then subtracted from Eq. A4.9 multiplying $\ln r_j$ to gives:

$$D_j\left[\ln\frac{r_{j-1}}{r_j}\right] = (\mathscr{E}_j \ln r_{j-1} - \mathscr{E}_{j-1}\ln r_j) + \frac{M_{j-1}}{I_j}\left[r_{j-1}^2\left[\frac{r_j}{4} - \frac{r_{j-1}}{9}\right]\ln r_j - r_j^2\left[\frac{r_j}{4} - \frac{r_j}{9}\right]\ln r_{j-1}\right] + \frac{M_j}{I_j}\left[r_{j-1}^2\left[\frac{r_{j-1}}{9} - \frac{r_{j-1}}{4}\right]\ln r_j - r_j^2\left[\frac{r_j}{9} - \frac{r_{j-1}}{4}\right]\ln r_{j-1}\right] \quad \text{(A4.13)}$$

$$D_j = \left\{(\mathscr{E}_j \ln r_{j-1} - \mathscr{E}_{j-1}\ln r_j) + \frac{M_{j-1}}{I_j}\left[r_{j-1}^2\left[\frac{r_j}{4} - \frac{r_{j-1}}{9}\right]\ln r_j - r_j^2\left[\frac{r_j}{4} - \frac{r_j}{9}\right]\ln r_{j-1}\right] + \frac{M_j}{I_j}\left[r_{j-1}^2\left[\frac{r_{j-1}}{9} - \frac{r_{j-1}}{4}\right]\ln r_j -\right.\right.$$

$$r_j^2\left(\frac{r_j}{9} - \frac{r_{j-1}}{4}\right)\ln r_{j-1}\Bigg] \Bigg\} \Big/ \left(\ln\frac{r_{j-1}}{r_j}\right) \qquad \text{(A4.14)}$$

The following constant are now defined:

$$\gamma_j = 1 \Big/ \left(\ln\frac{r_j}{r_{j-1}}\right) \qquad \text{(A4.15)}$$

$$\alpha_j = \left[r_j^2\left(\frac{r_{j-1}}{4} - \frac{r_j}{9}\right) - r_{j-1}^2\left(\frac{r_{j-1}}{4} - \frac{r_{j-1}}{9}\right)\right]\frac{\gamma_j}{l_j} \qquad \text{(A4.16)}$$

$$\beta_j = \left[r_j^2\left(\frac{r_j}{9} - \frac{r_j}{4}\right) - r_{j-1}^2\left(\frac{r_{j-1}}{9} - \frac{r_j}{4}\right)\right]\frac{\gamma_j}{l_j} \qquad \text{(A4.17)}$$

$$\gamma_{1j} = \ln r_{j-1} \Big/ \left(\ln\frac{r_{j-1}}{r_j}\right) \qquad \text{(A4.18)}$$

$$\gamma_{2j} = \ln r_j \Big/ \left(\ln\frac{r_{j-1}}{r_j}\right) \qquad \text{(A4.19)}$$

$$\alpha_{1j} = \left[\ln r_{j-1}\, r_j^2\left(\frac{r_{j-1}}{4} - \frac{r_j}{9}\right) - \ln r_j\, r_{j-1}^2\left(\frac{r_{j-1}}{4} - \frac{r_{j-1}}{9}\right)\right] \Big/ l_j\left(\ln\frac{r_{j-1}}{r_j}\right) \qquad \text{(A4.20)}$$

$$\beta_{1j} = \left[\ln r_{j-1}\, r_j^2\left(\frac{r_j}{9} - \frac{r_j}{4}\right) - \ln r_j\, r_{j-1}^2\left(\frac{r_{j-1}}{9} - \frac{r_j}{4}\right)\right] \Big/ l_j\left(\ln\frac{r_{j-1}}{r_j}\right) \qquad \text{(A4.21)}$$

All the above equation are valid for $j = 2,3,\ldots,j_m$.

At $j=1$, then:

$$\alpha_1 = \beta_1 = \gamma_1 = 0 \qquad \text{(A4.22)}$$

$$\alpha_{11} = \beta_{11} = \gamma_{11} = 0 \qquad \text{(A4.23)}$$

$$\gamma_{21} = -1. \qquad \text{(A4.24)}$$

By using Eqs. A.15 to A.21, Eq. A4.5 can be written as:

$$\mathscr{E} = \frac{M_{j-1}}{l_j} r^2\left(\frac{r_j}{4} - \frac{r}{9}\right) + \frac{M_j}{l_j} r^2\left(\frac{r}{9} - \frac{r_{j-1}}{4}\right) + \left[\alpha_j M_j + \beta_j M_{j-1} + \gamma_j(\mathscr{E}_j - \mathscr{E}_{j-1})\right]\ln r + \alpha_{1j} M_j + \beta_{1j} M_{j-1} + \gamma_{1j}\mathscr{E}_j - \gamma_{2j}\mathscr{E}_{j-1} \qquad \text{(A4.25)}$$

Differentiating Eq. A4.25 yields:

$$\frac{\partial \mathscr{E}}{\partial r} = M_j\left[\frac{\alpha_j}{r} + \frac{r}{l_j}\left[\frac{r}{3} - \frac{r_{j-1}}{4}\right]\right] + M_{j-1}\left[\frac{\beta_j}{r} + \frac{r}{l_j}\left[\frac{r_j}{2} - \frac{r}{3}\right]\right] + \gamma_j(\mathscr{E}_j - \mathscr{E}_{j-1})\Big] \quad \text{(A4.26)}$$

If $\frac{\partial \mathscr{E}}{\partial r}$ at $r=r_j$ is demanded to be continuous, then M_j must satisfy

$$A_jM_{j+1} + B_jM_j + C_jM_{j-1} = -\frac{\gamma_{j+1}}{r_j}(\mathscr{E}_{j+1} - \mathscr{E}_j) + \frac{\gamma_j}{r_j}(\mathscr{E}_j - \mathscr{E}_{j-1}) \quad \text{(A4.27)}$$

where A_j, B_j and C_j are defined by:

$$A_j = \frac{\alpha_{j+1}}{r_j} + \frac{r_j}{l_{j+1}}\left[\frac{r_j}{3} - \frac{r_j}{2}\right] \quad \text{(A4.28)}$$

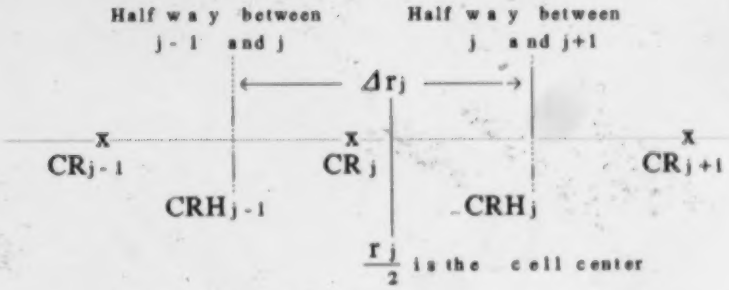
$$B_j = \frac{\beta_{j+1}}{r_j} + \frac{r_j}{l_{j+1}}\left[\frac{r_{j+1}}{2} - \frac{r_j}{3}\right] - \frac{\alpha_j}{r_j} - \frac{r_j}{l_j}\left[\frac{r_j}{3} - \frac{r_{j-1}}{2}\right] \quad \text{(A4.29)}$$

$$C_j = -\frac{\beta_j}{r_j} - \frac{r_j}{l_j}\left[\frac{r_j}{2} - \frac{r_j}{3}\right] \quad \text{(A4.30)}$$

APPENDIX 5

GRID STRUCTURE IN HEATER

Consider the r grid



where $\Delta r_j = CRH_j - CRH_{j-1}$

$$\frac{r_j}{2} = \frac{CRH_{j-1} + CRH_j}{2}$$

$$CRH_{j-1} = \frac{CR_{j-1} + CR_j}{2}$$

$$CRH_j = \frac{CR_j + CR_{j+1}}{2}$$

The quantity RDR integral of F around a mesh point can be approximated as

$$RDR_j = \frac{1}{2\pi F_j(\bar{r})}\int 2\pi F_j(r) r\, dr \simeq \int r dr \simeq \frac{r_j}{2}\Delta r_j$$

Therefore

$$RDR_j = \frac{1}{2}\left[CRH_j + CRH_{j-1} \right]\left[CRH_j - CRH_{j-1} \right]$$

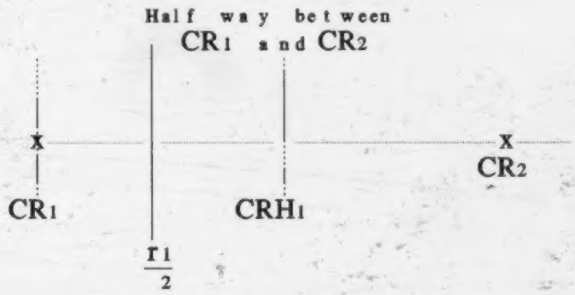
$$= \frac{1}{2}\left[\frac{CR_j + CR_{j+1}}{2} + \frac{CR_{j-1} + CR_j}{2}\right]\left[\frac{CR_j + CR_{j+1}}{2} - \frac{CR_{j-1} + CR_j}{2}\right]$$

$$= \frac{1}{2}\left[CR_j + \frac{1}{2}CR_{j+1} + \frac{1}{2}CR_{j-1} \right]\left[\frac{1}{2}CR_{j+1} - \frac{1}{2}CR_{j-1} \right]$$

$$= \frac{1}{4}CR_jCR_{j+1} - \frac{1}{4}CR_jCR_{j-1} + \frac{1}{8}CR_{j+1}^2 - \frac{1}{8}CR_{j-1}^2$$

$$= \frac{1}{8}\left[CR_{j+1}^2 - CR_{j-1}^2 \right] + \frac{1}{4}CR_j \left[CR_{j+1} - CR_{j-1} \right]$$

At the boundary, r = 0



$$\frac{r_1}{2} = \frac{CRH_1 + CR_1}{2}$$

$$\Delta r_1 = CRH_1 - CR_1$$

$$CRH_1 = \frac{CR_1 + CR_2}{2}$$

Therefore, at the boundary (r = 0)

$$RDR_1 = \frac{1}{2\pi F_1(r)} \int 2\pi F_1(r) r \, dr \simeq \int r dr \simeq \frac{r_1}{2}\Delta r_1$$

$$= \left[\frac{CRH_1 + CR_1}{2} \right]\left[CRH_1 - CR_1 \right]$$

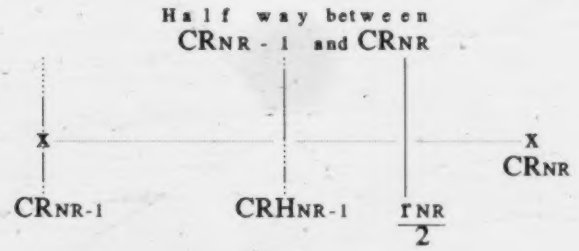
$$= \left[\frac{CR_1}{2} + \frac{1}{2}\frac{CR_1 + CR_2}{2} \right]\left[\frac{CR_1 + CR_2}{2} - CR_1 \right]$$

$$= \left[\frac{3}{4}CR_1 + \frac{1}{4}CR_2 \right]\left[\frac{1}{2}CR_1 + \frac{1}{2}CR_2 \right]$$

$$= -\frac{3}{8}CR_1^2 + \frac{3}{8}CR_1CR_2 - \frac{1}{8}CR_1CR_2 + \frac{1}{8}CR_2^2$$

$$= \frac{1}{8}\left[-3CR_1^2 + CR_2^2 \right] + \frac{1}{4}CR_1CR_2$$

At the boundary r = R_N



$$RDR_{NR} = \frac{1}{2\pi F_{NR}(r)} \int 2\pi F_{NR}(r) r \, dr \simeq \int r dr \simeq \frac{r_{NR}}{2} \Delta r_{NR}$$

$$\frac{r_{NR}}{2} = \frac{CRH_{NR-1} + CR_{NR}}{2}$$

$$\Delta r_{NR} = CR_{NR} - CRH_{NR-1}$$

$$CRH_{NR-1} = \frac{CR_{NR-1} + CR_{NR}}{2}$$

Therefore

$$RDR_{NR} = \left[\frac{CRH_{NR-1} + CR_{NR}}{2} \right] \left[CR_{NR} - CRH_{NR-1} \right]$$

$$= \left[\frac{CR_{NR}}{2} + \frac{CR_{NR-1} + CR_{NR}}{2} \right] \left[CR_{NR} - \frac{CR_{NR-1} + CR_{NR}}{2} \right]$$

$$= \left[\frac{3}{4} CR_{NR} + \frac{1}{4} CR_{NR-1} \right] \left[\frac{1}{2} CR_{NR} - \frac{1}{2} CR_{NR-1} \right]$$

$$= \frac{1}{8} \left[3CR_{NR}^2 - CR_{NR-1}^2 \right] - \frac{1}{4} CR_{NR} CR_{NR-1}$$

TABLE 1

Vector operation time for CYBER 205

Operation	Time 64 bit		Time 32 bit	
	2 pipes	4 pipes	2 pipes	4 pipes
Add/Subtract	51+N/2	51+N/4	51+N/4	51+N/8
Multiply	52+N/2	52+N/4	52+N/4	52+N/8
Link Vector add and mult. triad	103+N/2	103+N/4	103+N/4	103+N/8
Divide	80+N/.28	80+N/.56	68+N/.5	68+N
Square root	79+N/.28	79+N/.56	67+N/.5	67+N
Scatter vector randomly	83+N/.8	83+N/.8	83+N/.8	83+N/.8
Scatter vector periodically	71+N/.8	71+N/.8	71+N/.8	71+N/.8
Gather vector randomly	69+N/.8	69+N/.8	69+N/.8	69+N/.8
Gather vector periodically	39+N/.8	39+N/.8	39+N/.8	39+N/.8
Compress vector A (Z # of elem. comp.).	52+Z/2	52+Z/4	52+Z/4	52+Z/8
Expand vector A (z # of elem. Expa.)	58+Z/2	58+Z/4	52+Z/4	58+Z/8
Sum of elements	116+N	116+N	116+N	116+N
Dot product	116+N	116+N	116+N	116+N

Vector length = N

Total time = startup time + result rate.

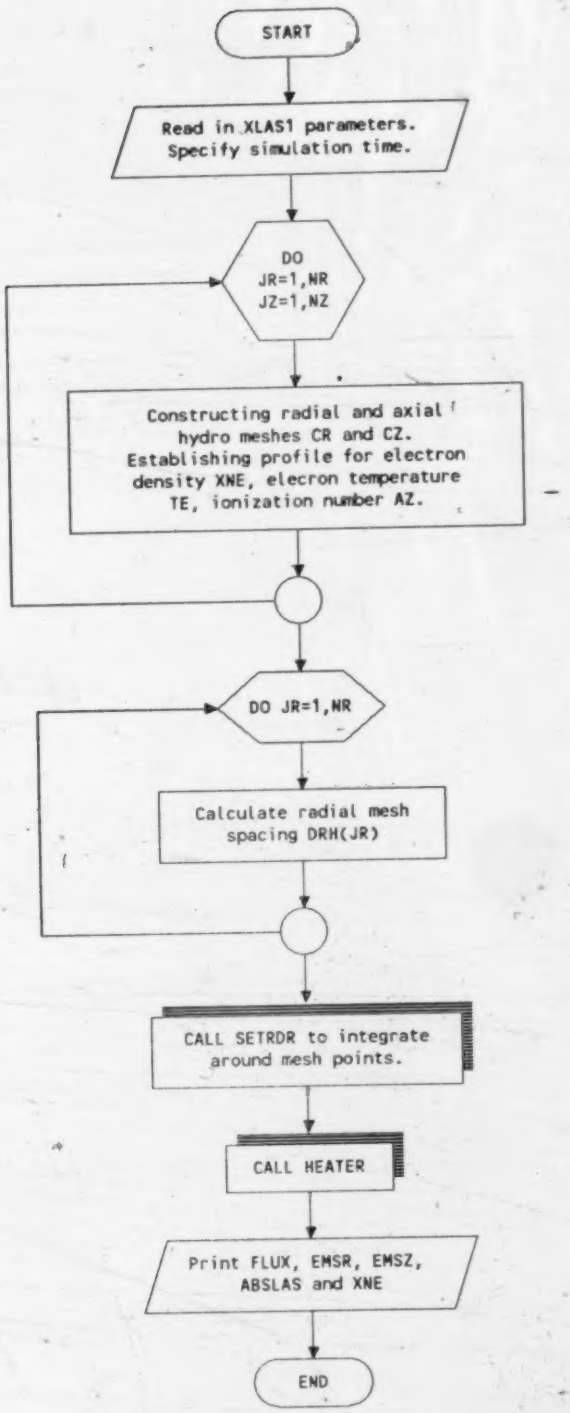
TABLE 2

XLAS1 ARRAY DEFINITION

ELEMENTS	DEFINITION
XLAS1(1)	Time laser switch on.
XLAS1(2)	Time laser switch off.
XLAS1(3)	Definition of pump,
XLAS1(4)	intensity, times,
XLAS1(5)	rise time,...
XLAS1(6)	according to pulse types.
XLAS1(7)	Radial fine mesh spacing.
XLAS1(8)	Determine laser pulse types.
XLAS1(9)	Control switch: laser on (+1) / laser off (-1).
XLAS1(10)	Total energy in the pulse.
XLAS1(11)	Number of radial meshes extended outside plasma.
XLAS1(12)	Spot size of the beam σ_0.
XLAS1(13)	Position on distance of focal length.
XLAS1(14)	Incoherency parameter.
XLAS1(15)	Maximum number of subdivision in z.
XLAS1(16)	Set to 1 if HEATER is to call, 0 otherwise.
XLAS1(17)	Max fraction change in power befor Heater is called.
XLAS1(18)	Max time before a call to HEATER
XLAS1(19)	Determine laser pulse shapes.
XLAS1(20)	Not used.

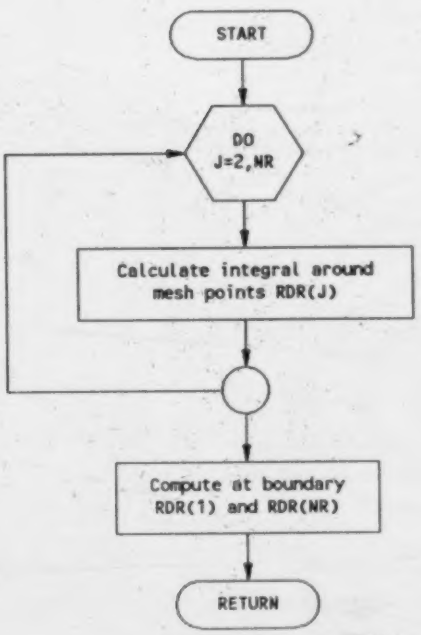
FLOW CHART 1

TEST HEATER


START
Read in XLAS1 parameters. Specify simulation time.
DO JR=1,NR JZ=1,NZ
Constructing radial and axial hydro meshes CR and CZ. Establishing profile for electron density XNE, elecron temperature TE, ionization number AZ.
DO JR=1,NR
Calculate radial mesh spacing DRH(JR)
CALL SETRDR to integrate around mesh points.
CALL HEATER
Print FLUX, EMSR, EMSZ, ABSLAS and XNE
END

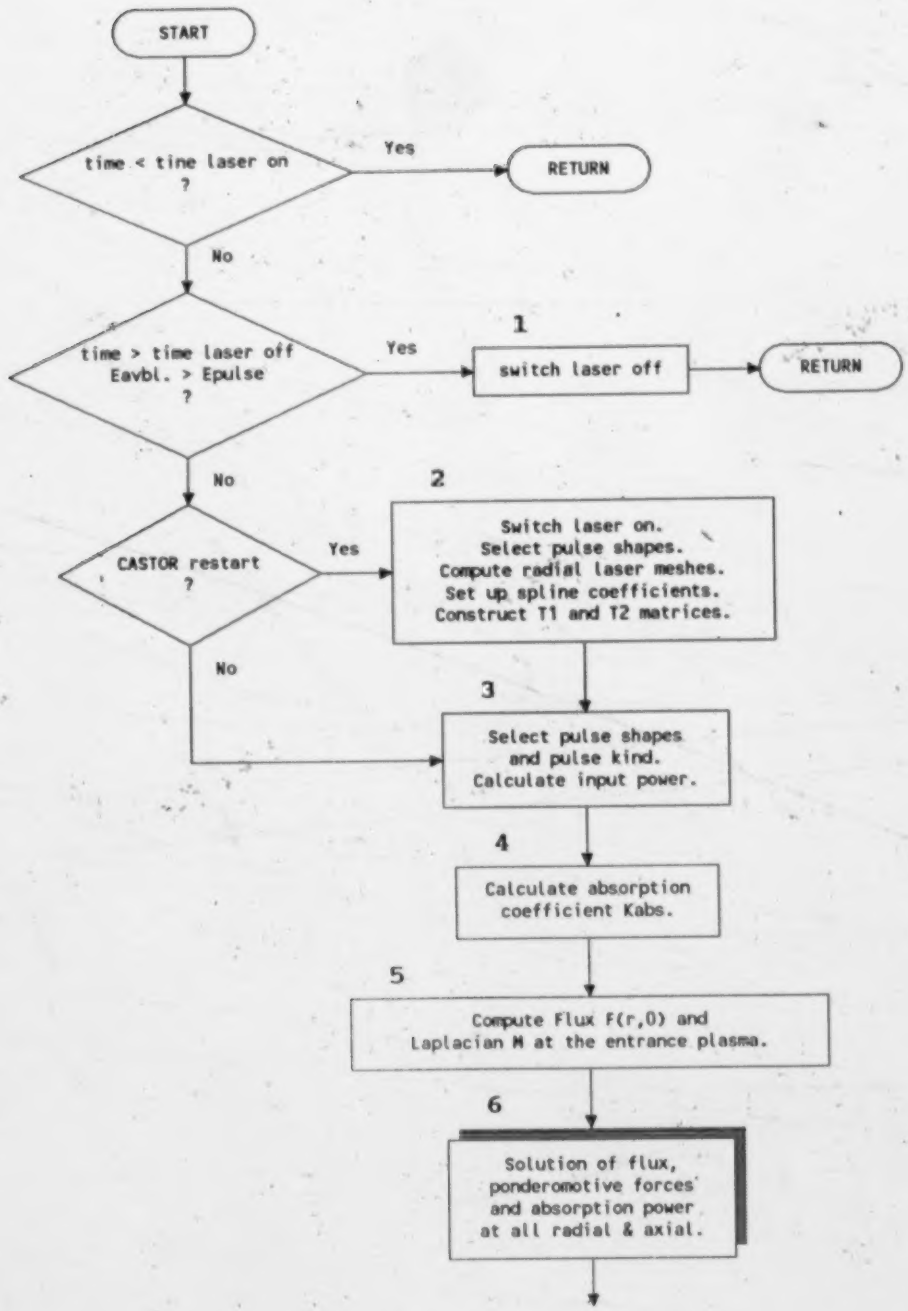
FLOW CHART 2

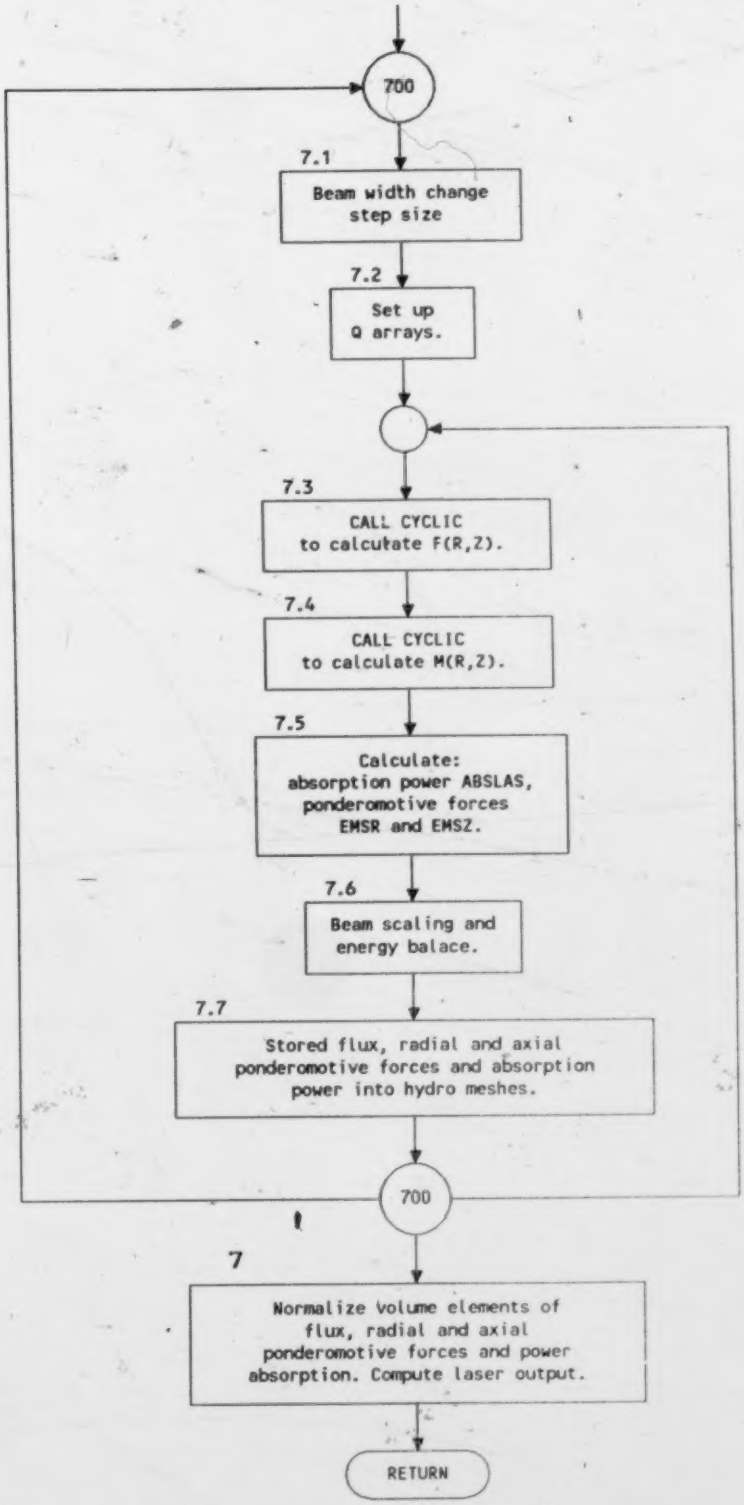
SUBROUTINE SETRDR



FLOW CHART 3

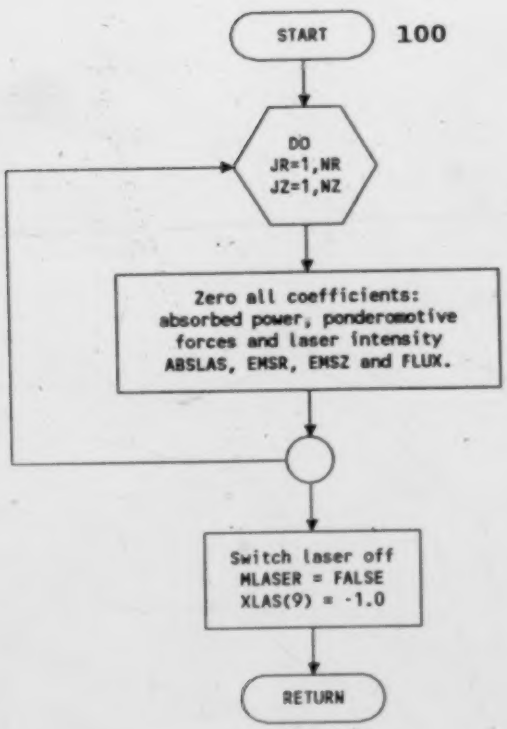
SUBROUTINE HEATER




700
7.1
Beam width change step size
7.2
Set up Q arrays.
7.3
CALL CYCLIC to calculate F(R,Z).
7.4
CALL CYCLIC to calculate M(R,Z).
7.5
Calculate: absorption power ABSLAS, ponderomotive forces EMSR and EMSZ.
7.6
Beam scaling and energy balace.
7.7
Stored flux, radial and axial ponderomotive forces and absorption power into hydro meshes.
700
7
Normalize volume elements of flux, radial and axial ponderomotive forces and power absorption. Compute laser output.
RETURN

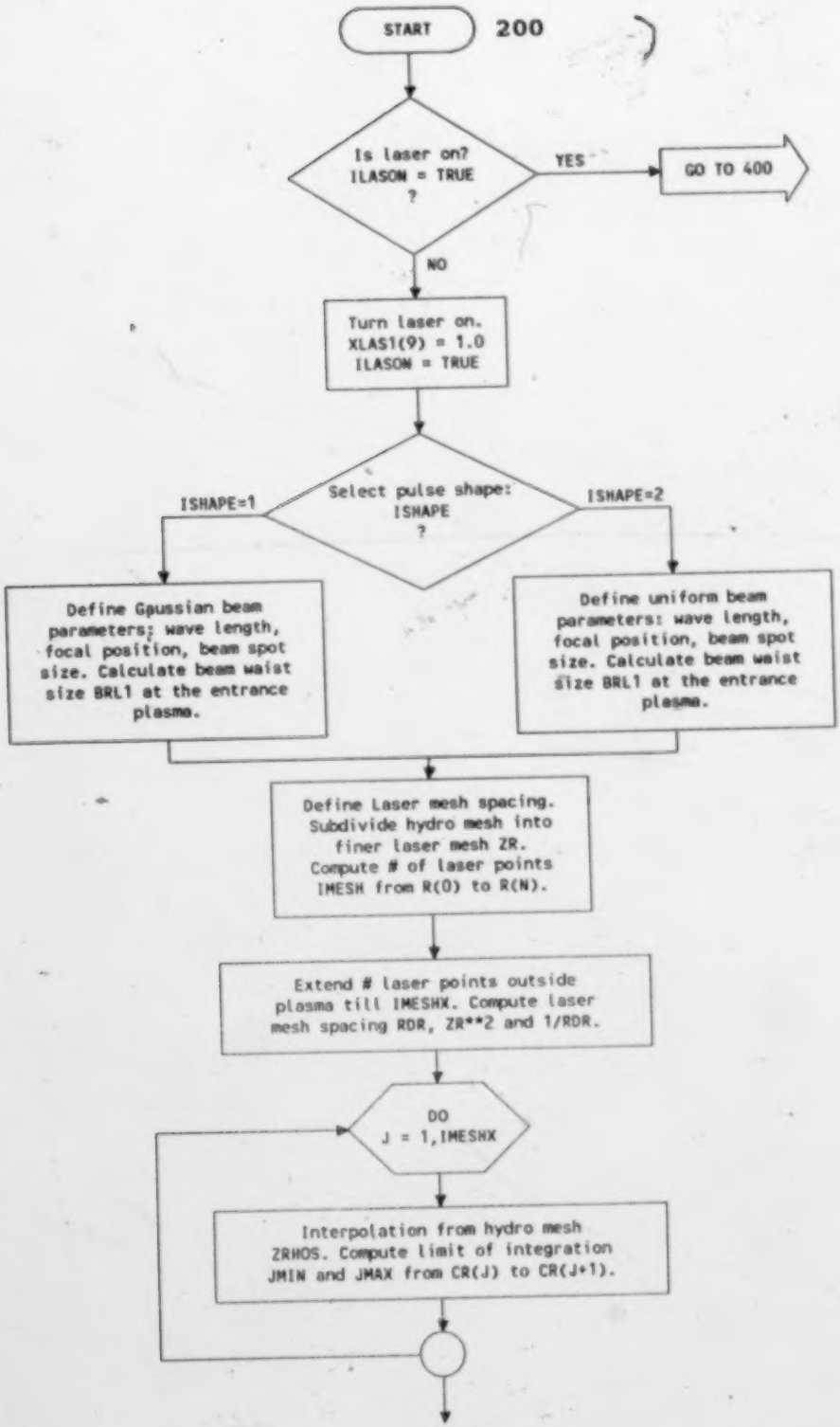
FLOW CHART 4

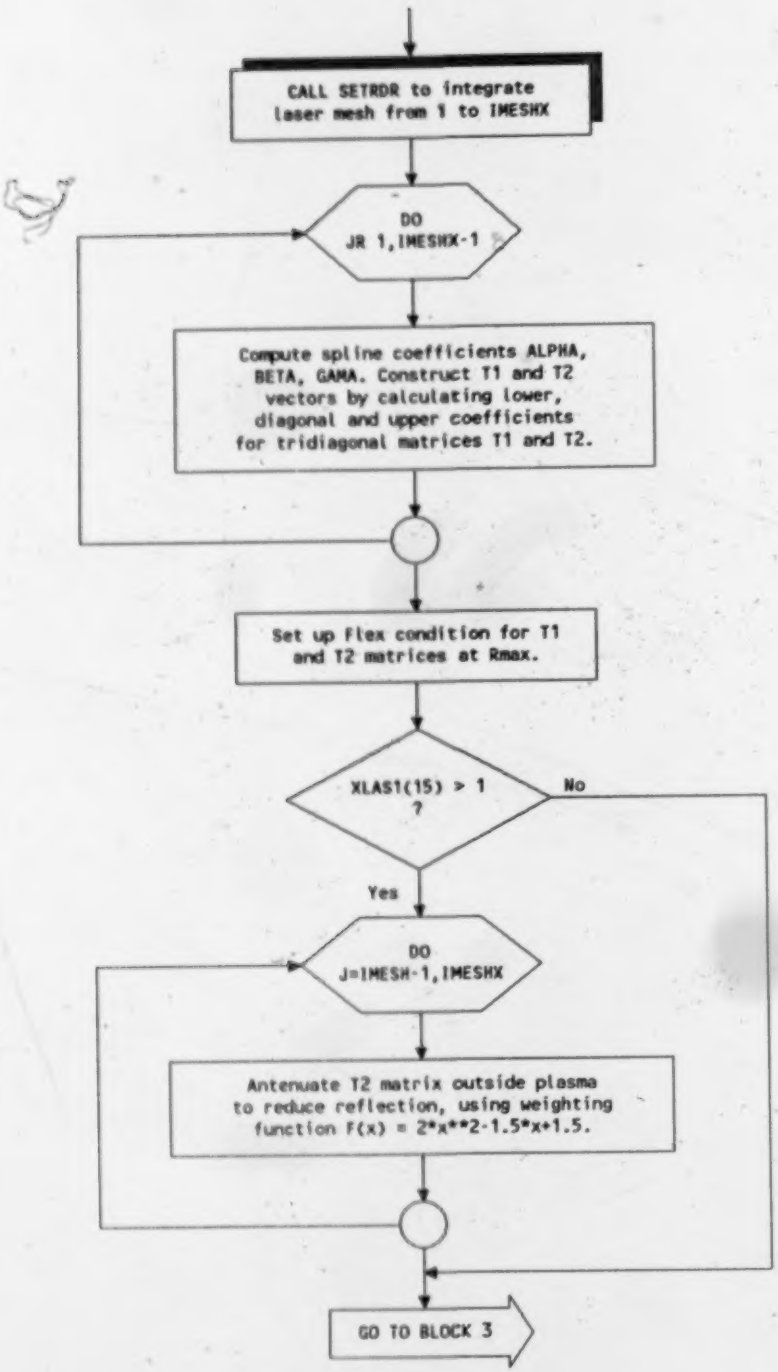
BLOCK 1



FLOW CHART 5

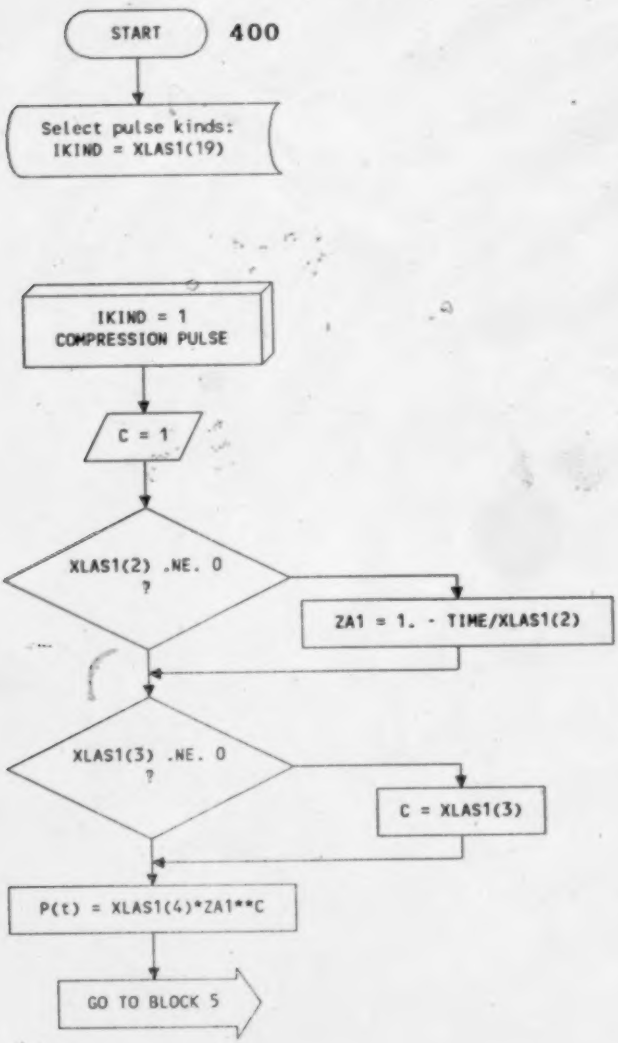
BLOCK 2

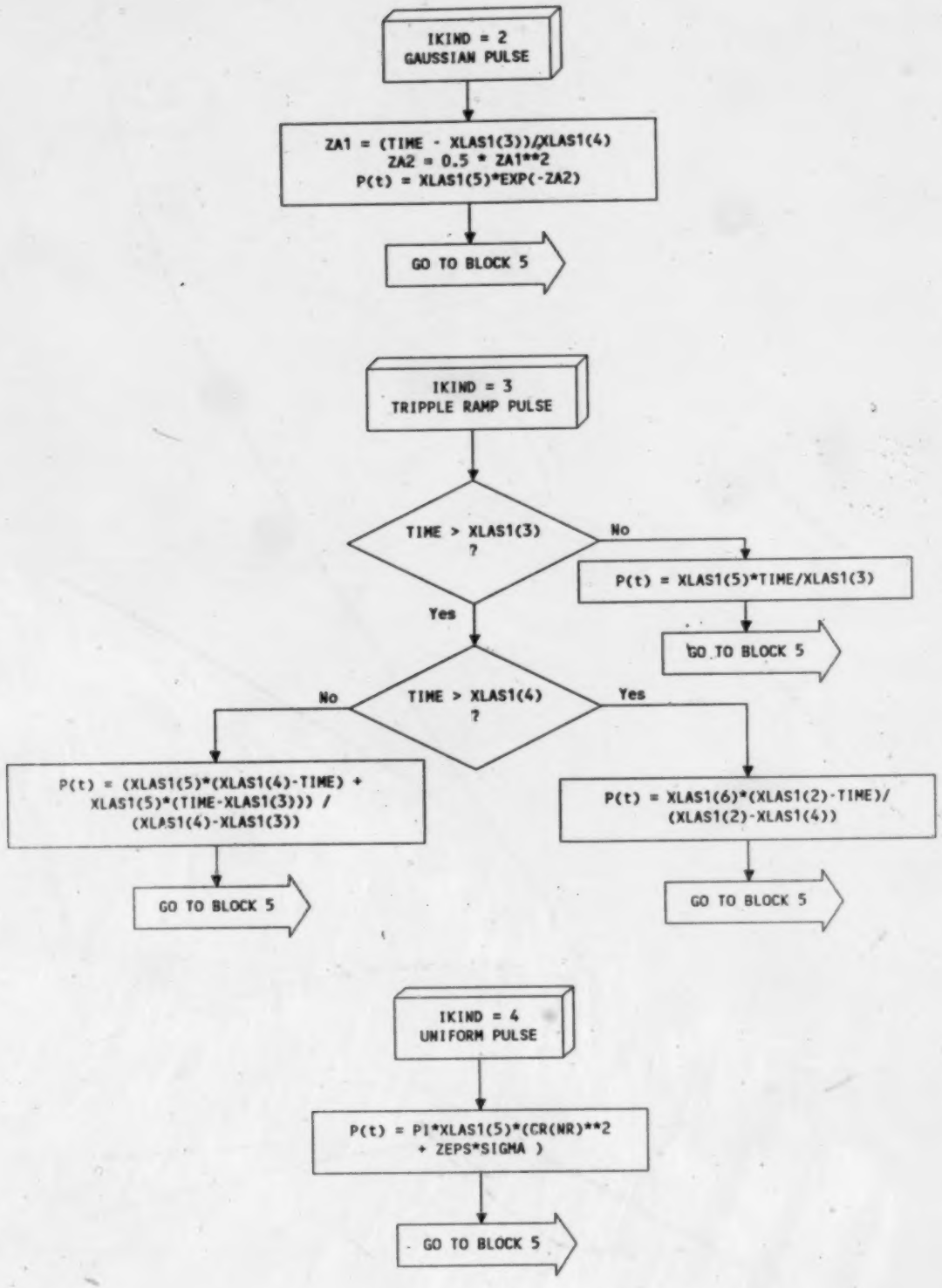

START
200
Is laser on?
ILASON = TRUE
?
YES
GO TO 400
NO
Turn laser on.
XLAS1(9) = 1.0
ILASON = TRUE
Select pulse shape:
ISHAPE
?
ISHAPE=1
ISHAPE=2
Define Gaussian beam parameters: wave length, focal position, beam spot size. Calculate beam waist size BRL1 at the entrance plasma.
Define uniform beam parameters: wave length, focal position, beam spot size. Calculate beam waist size BRL1 at the entrance plasma.
Define Laser mesh spacing. Subdivide hydro mesh into finer laser mesh ZR. Compute # of laser points IMESH from R(0) to R(N).
Extend # laser points outside plasma till IMESHX. Compute laser mesh spacing RDR, ZR**2 and 1/RDR.
DO
J = 1,IMESHX
Interpolation from hydro mesh ZRHOS. Compute limit of integration JMIN and JMAX from CR(J) to CR(J+1).


CALL SETRDR to integrate laser mesh from 1 to IMESHX
DO JR 1,IMESHX-1
Compute spline coefficients ALPHA, BETA, GAMA. Construct T1 and T2 vectors by calculating lower, diagonal and upper coefficients for tridiagonal matrices T1 and T2.
Set up Flex condition for T1 and T2 matrices at Rmax.
XLAS1(15) > 1 ?
No
Yes
DO J=IMESH-1,IMESHX
Antenuate T2 matrix outside plasma to reduce reflection, using weighting function F(x) = 2*x**2-1.5*x+1.5.
GO TO BLOCK 3

FLOW CHART 6

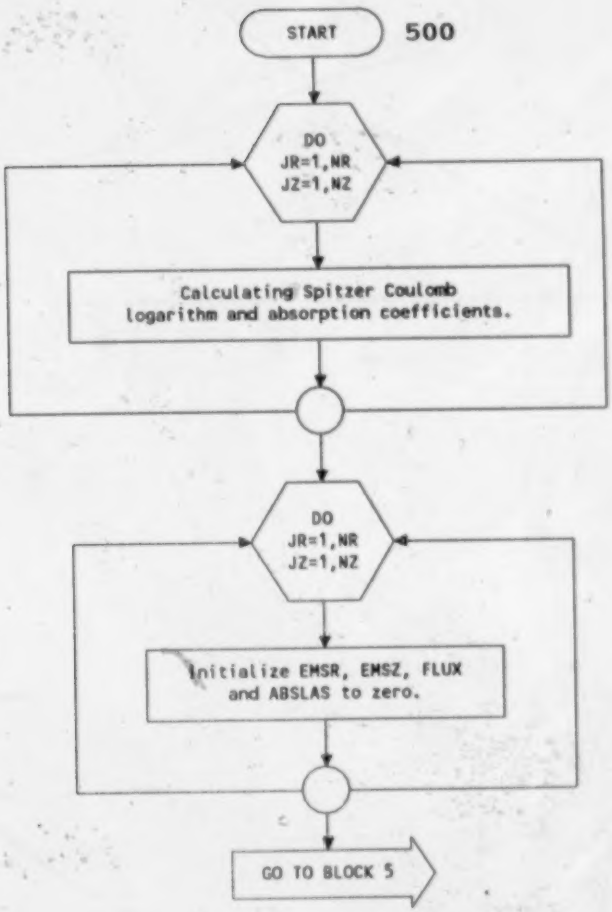
BLOCK 3




IKIND = 2
GAUSSIAN PULSE
ZA1 = (TIME - XLAS1(3))/XLAS1(4)
ZA2 = 0.5 * ZA1**2
P(t) = XLAS1(5)*EXP(-ZA2)
GO TO BLOCK 5
IKIND = 3
TRIPPLE RAMP PULSE
TIME > XLAS1(3)
?
No
P(t) = XLAS1(5)*TIME/XLAS1(3)
GO TO BLOCK 5
Yes
TIME > XLAS1(4)
?
No
Yes
P(t) = (XLAS1(5)*(XLAS1(4)-TIME) +
XLAS1(5)*(TIME-XLAS1(3))) /
(XLAS1(4)-XLAS1(3))
GO TO BLOCK 5
P(t) = XLAS1(6)*(XLAS1(2)-TIME)/
(XLAS1(2)-XLAS1(4))
GO TO BLOCK 5
IKIND = 4
UNIFORM PULSE
P(t) = PI*XLAS1(5)*(CR(NR)**2
+ ZEPS*SIGMA)
GO TO BLOCK 5

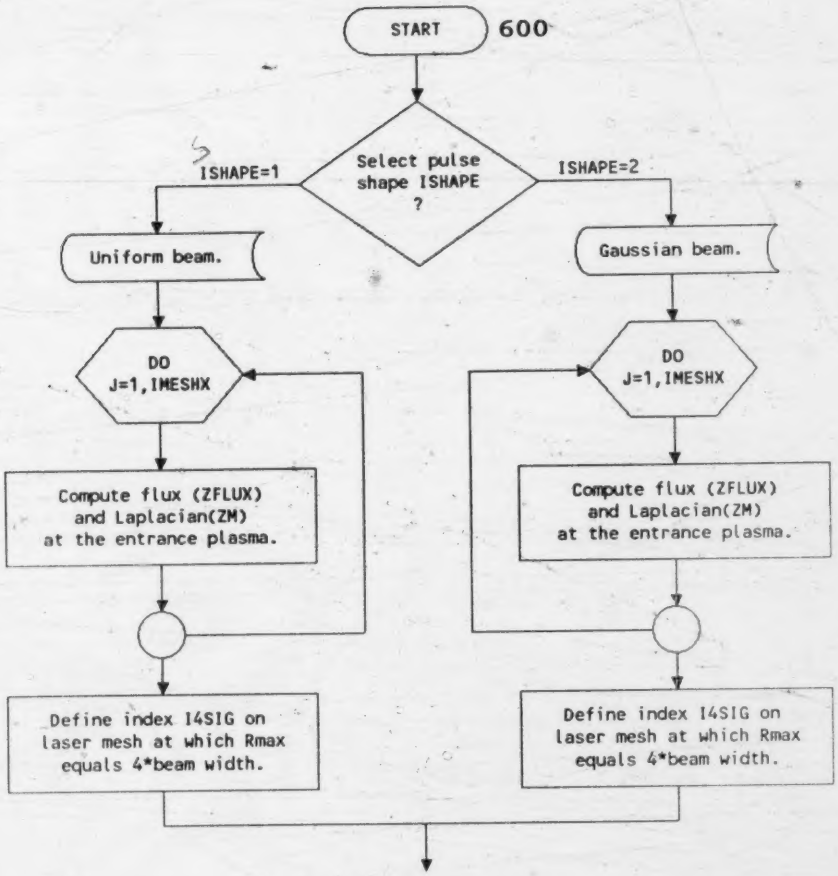
FLOW CHART 7

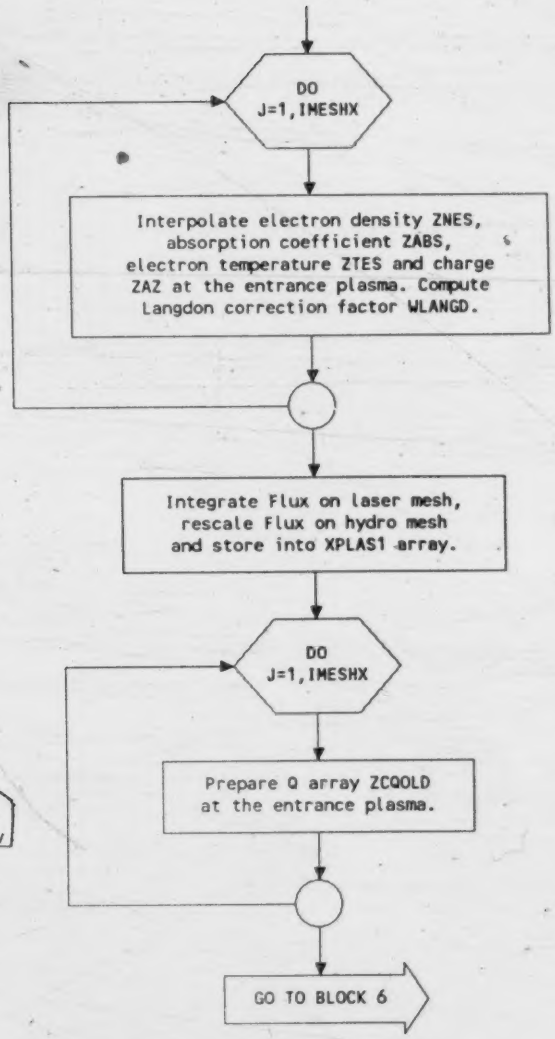
BLOCK 4



FLOW CHART 8

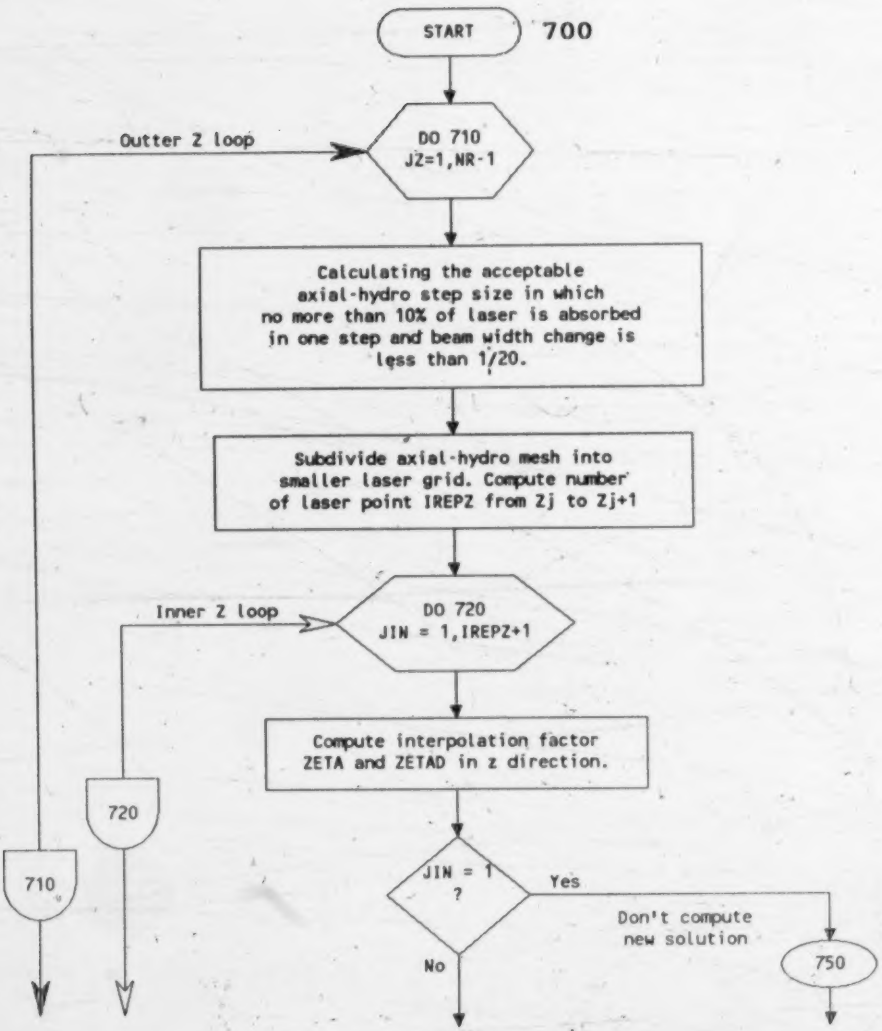
BLOCK 5

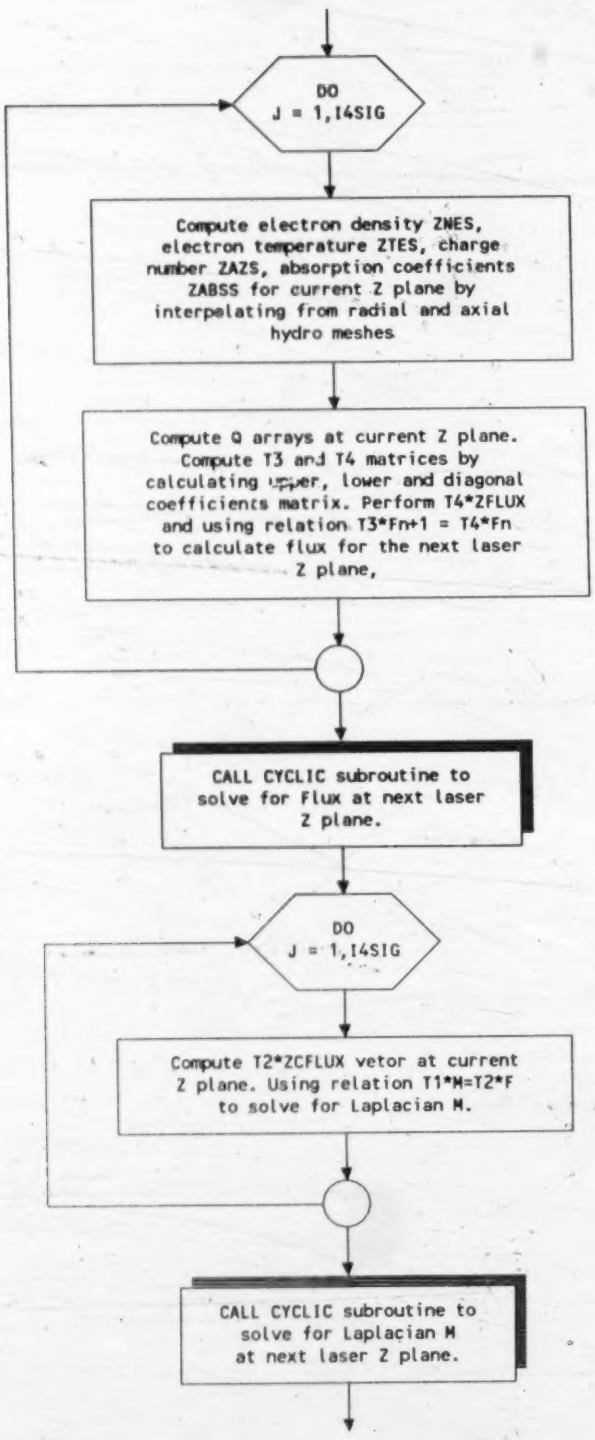

START
600
Select pulse shape ISHAPE ?
ISHAPE=1
ISHAPE=2
Uniform beam.
Gaussian beam.
DO J=1,IMESHX
DO J=1,IMESHX
Compute flux (ZFLUX) and Laplacian(ZM) at the entrance plasma.
Compute flux (ZFLUX) and Laplacian(ZM) at the entrance plasma.
Define index I4SIG on laser mesh at which Rmax equals 4*beam width.
Define index I4SIG on laser mesh at which Rmax equals 4*beam width.

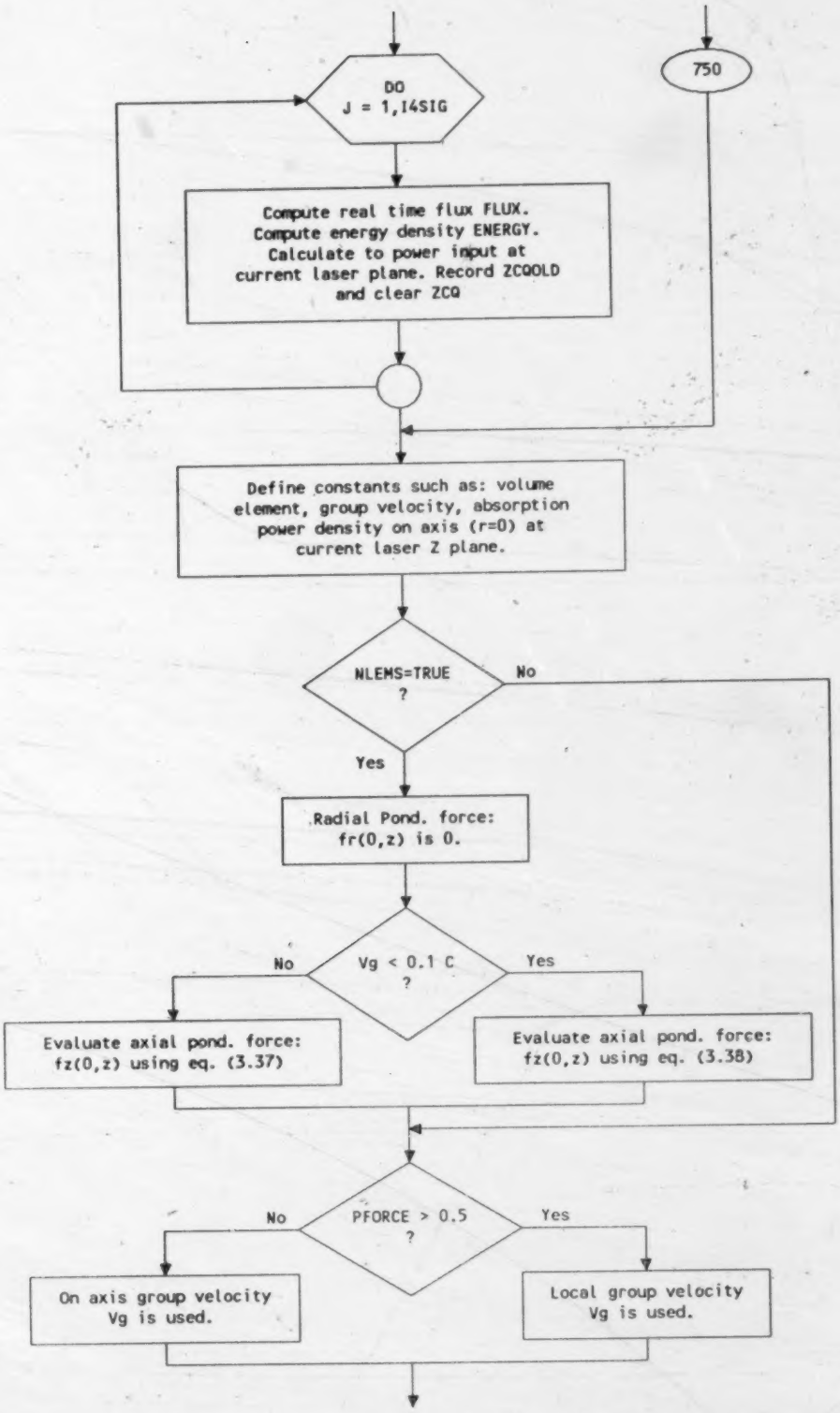

DO
J=1,IMESHX
Interpolate electron density ZNES, absorption coefficient ZABS, electron temperature ZTES and charge ZAZ at the entrance plasma. Compute Langdon correction factor WLANGD.
Integrate Flux on laser mesh, rescale Flux on hydro mesh and store into XPLAS1 array.
DO
J=1,IMESHX
Prepare Q array ZCQOLD at the entrance plasma.
GO TO BLOCK 6

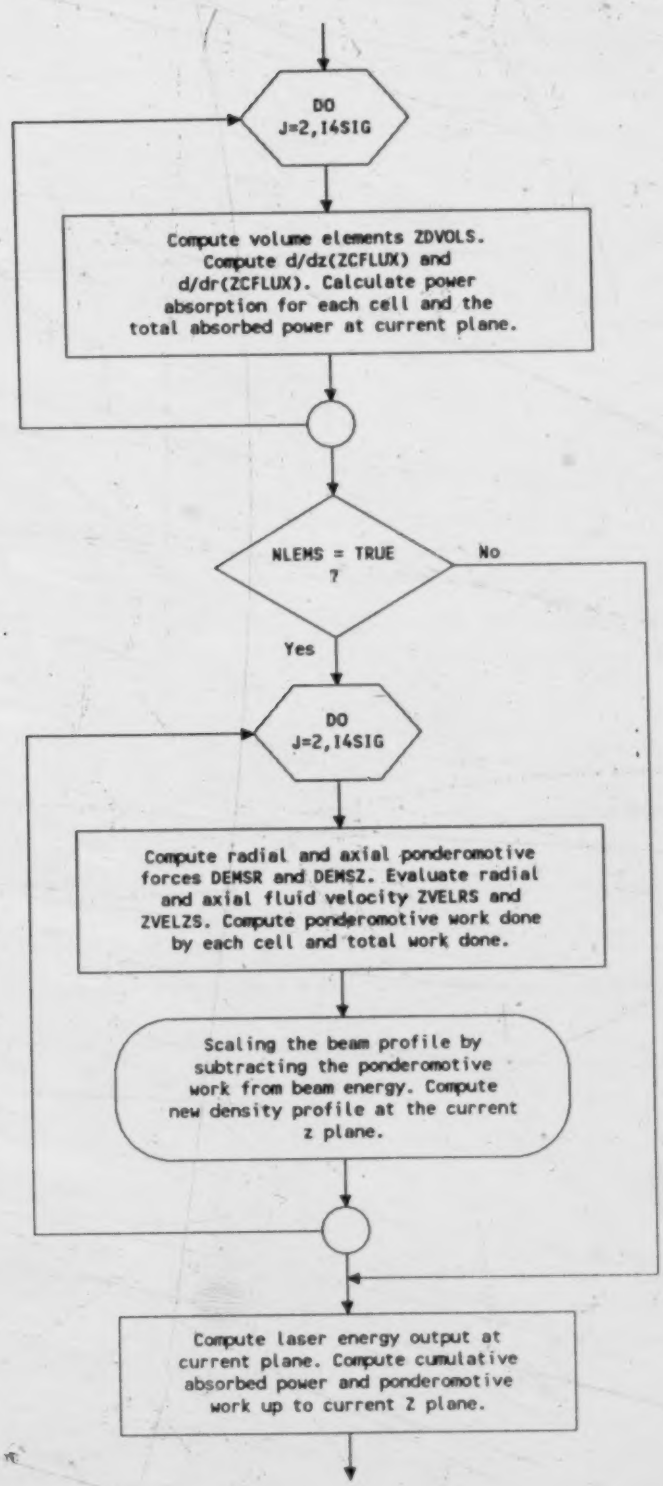
FLOW CHART 9

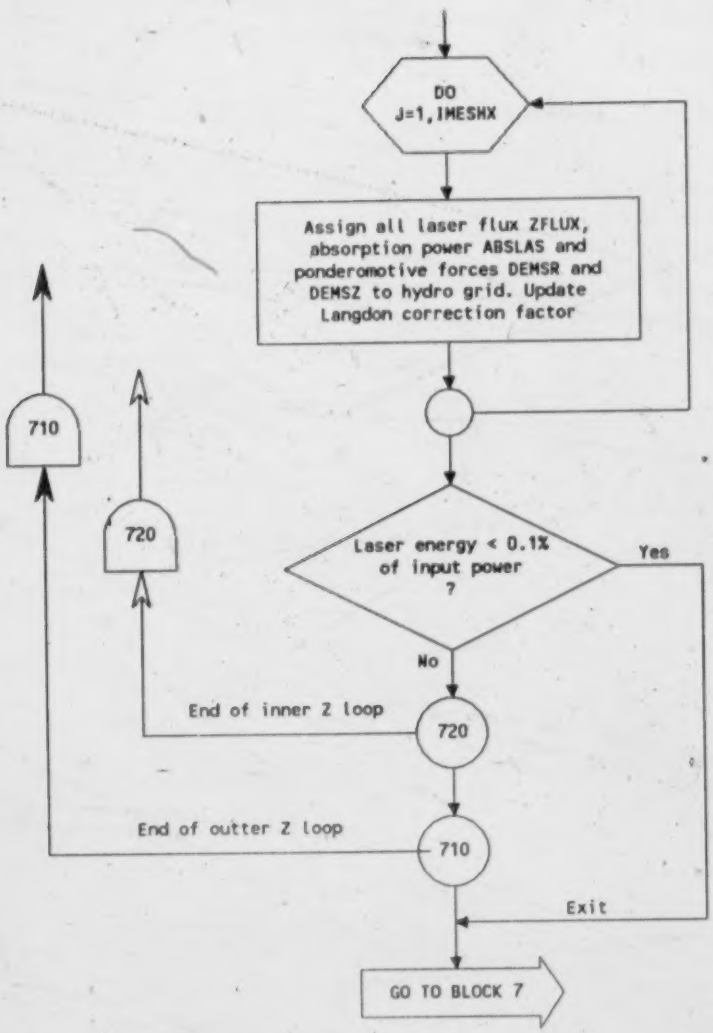
BLOCK 6




DO
J = 1,I4SIG
Compute electron density ZNES, electron temperature ZTES, charge number ZAZS, absorption coefficients ZABSS for current Z plane by interpolating from radial and axial hydro meshes
Compute Q arrays at current Z plane. Compute T3 and T4 matrices by calculating upper, lower and diagonal coefficients matrix. Perform T4*ZFLUX and using relation T3*Fn+1 = T4*Fn to calculate flux for the next laser Z plane,
CALL CYCLIC subroutine to solve for Flux at next laser Z plane.
DO
J = 1,I4SIG
Compute T2*ZCFLUX vetor at current Z plane. Using relation T1*M=T2*F to solve for Laplacian M.
CALL CYCLIC subroutine to solve for Laplacian M at next laser Z plane.

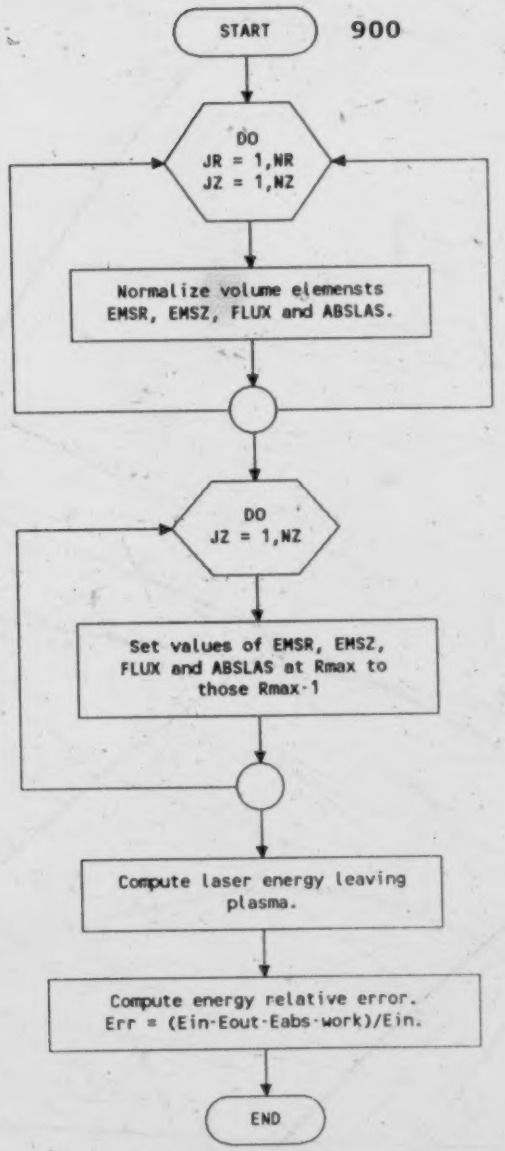

DO
J = 1,I4SIG
750
Compute real time flux FLUX.
Compute energy density ENERGY.
Calculate to power input at
current laser plane. Record ZCQOLD
and clear ZCQ
Define constants such as: volume
element, group velocity, absorption
power density on axis (r=0) at
current laser Z plane.
NLEMS=TRUE
?
No
Yes
Radial Pond. force:
fr(0,z) is 0.
Vg < 0.1 C
?
No
Yes
Evaluate axial pond. force:
fz(0,z) using eq. (3.37)
Evaluate axial pond. force:
fz(0,z) using eq. (3.38)
PFORCE > 0.5
?
No
Yes
On axis group velocity
Vg is used.
Local group velocity
Vg is used.


DO
J=2,14SIG
Compute volume elements ZDVOLS. Compute d/dz(ZCFLUX) and d/dr(ZCFLUX). Calculate power absorption for each cell and the total absorbed power at current plane.
NLEMS = TRUE ?
No
Yes
DO
J=2,14SIG
Compute radial and axial ponderomotive forces DEMSR and DEMSZ. Evaluate radial and axial fluid velocity ZVELRS and ZVELZS. Compute ponderomotive work done by each cell and total work done.
Scaling the beam profile by subtracting the ponderomotive work from beam energy. Compute new density profile at the current z plane.
Compute laser energy output at current plane. Compute cumulative absorbed power and ponderomotive work up to current Z plane.


DO
J=1,IMESHX
Assign all laser flux ZFLUX, absorption power ABSLAS and ponderomotive forces DEMSR and DEMSZ to hydro grid. Update Langdon correction factor
710
720
Laser energy < 0.1% of input power ?
Yes
No
End of inner Z loop
720
End of outter Z loop
710
Exit
GO TO BLOCK 7

FLOW CHART 10

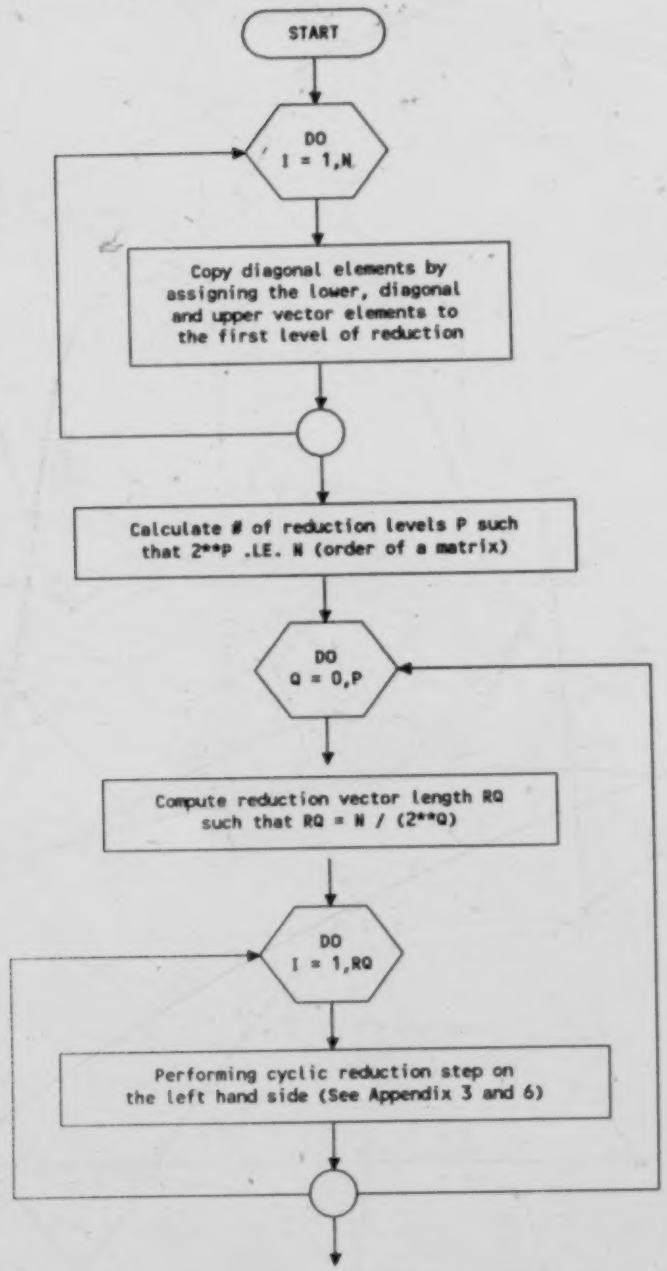
BLOCK 7

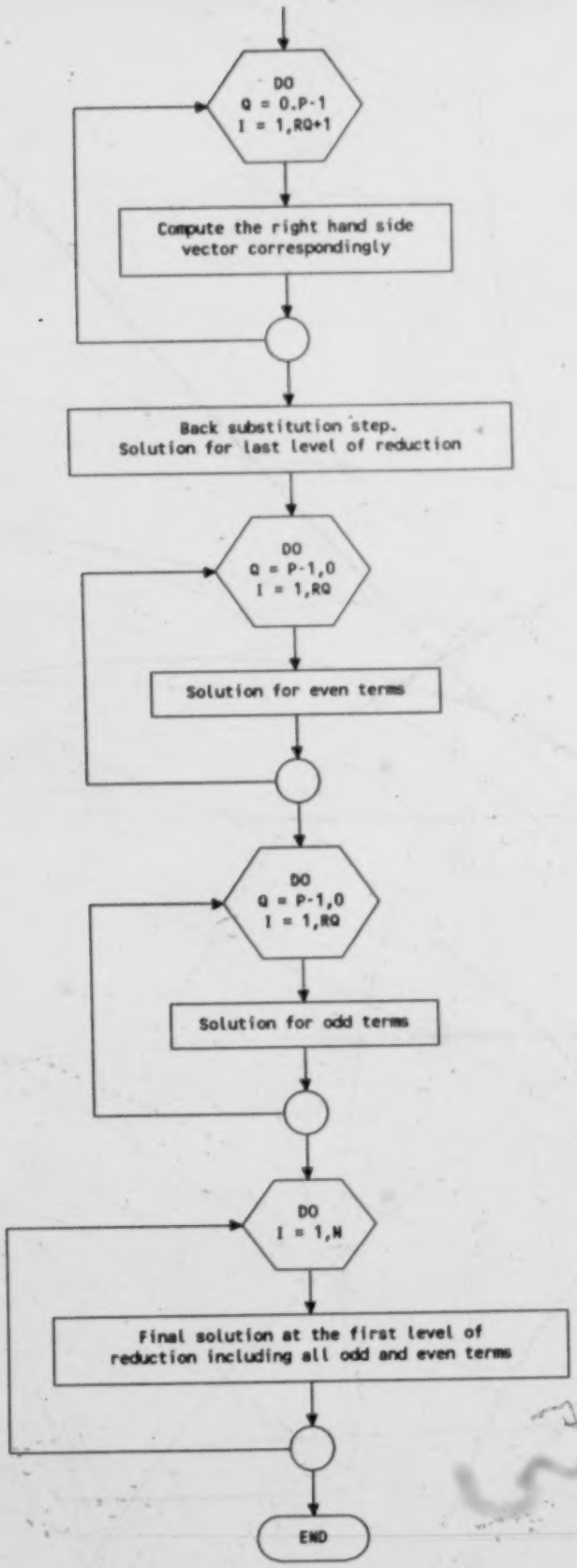


End of HEATER

FLOW CHART 11

SUBROUTINE CYCLIC


START
DO
I = 1,N
Copy diagonal elements by assigning the lower, diagonal and upper vector elements to the first level of reduction
Calculate # of reduction levels P such that 2**P .LE. N (order of a matrix)
DO
Q = 0,P
Compute reduction vector length RQ such that RQ = N / (2**Q)
DO
I = 1,RQ
Performing cyclic reduction step on the left hand side (See Appendix 3 and 6)


DO
Q = 0,P-1
I = 1,RQ+1
Compute the right hand side vector correspondingly
Back substitution step.
Solution for last level of reduction
DO
Q = P-1,0
I = 1,RQ
Solution for even terms
DO
Q = P-1,0
I = 1,RQ
Solution for odd terms
DO
I = 1,N
Final solution at the first level of reduction including all odd and even terms
END

```
C-----------------------------------------------------------------
C *** HEATER HEATER HEATER HEATER HEATER HEATER HEATER ***
C-----------------------------------------------------------------
C
      SUBROUTINE HEATER
C
C     SOLVE BEAM PROPAGATION EQUATION BY CUBIC SPLINE METHOD
C     CECA,RANK,JIMO,CCBA,DANG - ALBERTA
C     THIS SUBROUTINE IS A IMPLICIT VECTORIZABLE CODE WHICH
C     CAN BE RUN ON THE CRAY-XMP & CYBER 205 SUPERCOMPUTERS
C
        PARAMETER (NFMMAX=300,MAXDIZ=7,NREXT=60)
        PARAMETER (CLIGHT=3.0E8,CTENTH=3.0E7)
C
C  INSERT COMMON BLOCKS PASSED FROM HYDRO CODE (IE. CASTOR)
C
C/ INSERT PARAM
C/ INSERT COMBAS
C/ INSERT COMCAS
C/ INSERT COMLAS
C/ INSERT COMESH
C/ INSERT COMFUN
C/ INSERT COMNC
C/ INSERT COMCNC
C/ INSERT COMTE
C/ INSERT COMAZ
C/ INSERT COMEQ
C/ INSERT COMEN
C/ INSERT COMVEL
C/ INSERT COMDT
C-----------------------------------------------------------------
      EQUIVALENCE(RBGEN,PFORCE)
C
      REAL ZRDR(NFMMAX),  WLANGD(NFMMAX), DABSLA(NFMMAX),
     +     DEMSR(NFMMAX), DEMSZ(NFMMAX),  ZR(NFMMAX),
     +     ZDR(NFMMAX),   ZALFA(NFMMAX),  ZBETA(NFMMAX),
     +     ZGAMA(NFMMAX), ZT1(NFMMAX,3),  ZT2(NFMMAX,3),
     +     ZFLUX(NFMMAX), ZABS(NRMAXI,NZMAXI)
C
      COMPLEX ZCFAC,ZCEXP,ZNFAC,ZCK,
     +        ZQ0,ZC1,ZDEDZ, ZCQOLD(NFMMAX),
     +        ZCFLUX(NFMMAX), ZCQ(NFMMAX), ZM(NFMMAX)
C
C-----------------------------------------------------------------
C      DECLARE ARRAYS USED TO AID VECTORIZATION
C-----------------------------------------------------------------
C
      INTEGER JRS(NFMMAX),   JRP1S(NFMMAX), JMINS(NRMAXI),
     +        JMAXS(NRMAXI), IRRS(NFMMAX),  IBCR(NRMAXI),
     +        IREPR(NRMAXI)
C
      REAL TEMPS(NFMMAX),  TEMP1(NFMMAX),  ZRQS(NFMMAX),
```

```
53        +      ZRRS(NFMMAX),    ZDRRS(NFMMAX),   ZRHOS(NFMMAX),
54        +      ZRHODS(NFMMAX), ZNES(NFMMAX),    ZNEOS(NFMMAX),
55        +      ZTES(NFMMAX),    ZAZS(NFMMAX),    ZABSS(NFMMAX),
56        +      ZVGOS(NFMMAX),   ZA1S(NFMMAX),    ZAS(NFMMAX),
57        +      ZKRS(NFMMAX),    ZQREAS(NFMMAX), ZQIMAS(NFMMAX),
58        +      ZNERS(NFMMAX),   ZNRMS(NFMMAX),   ZDNDZS(NFMMAX),
59        +      ZVGS(NFMMAX),    ZDVOLS(NFMMAX), ZIMAGS(NFMMAX),
60        +      ZKA1S(NFMMAX),   ZVELRS(NFMMAX), ZVELZS(NFMMAX),
61        +      WORKR(NFMMAX),   WORKZ(NFMMAX)
62  C
63         COMPLEX ZCT1(NFMMAX,3), CZE(NFMMAX),     CZT3(NFMMAX,3),
64        +        CZKMQ(NFMMAX),  CZKPQ(NFMMAX),  ZDEDZS(NFMMAX),
65        +        ZDEDRS(NFMMAX), ZCONJS(NFMMAX), ZDS(NFMMAX),
66        +        ZCQS(NFMMAX),   ZDMS(NFMMAX),   CZT1P(NFMMAX,3),
67        +        ZQRZS
68  C
69  C-------------------------------------------------------------
70  C
71        LOGICAL ILASON
72  C
73        DATA ILASON,PWRAB1,PWRPO1/.FALSE.,0.0,0.0/
74        DATA ZTWOPI,ISHORT,ACUMT/6.283185,0,0.0/
75  C
76  C  PRECALCULATE VALUES SUCH AS SQUARE OF WAVELENGTH,
77  C  SPEED OF LIGHT, 0.99*CRITICAL DENSITY ETC.
78  C  WHICH ARE USED THROUGHOUT THE RUN
79  C
80        XL1Q    = XL1*XL1
81        XNC1F   = 0.99*XNC1
82        XNC1R   = 1.0/XNC1
83        XL1QF   = 0.464*XL1Q
84        CLIGHQ  = CLIGHT*CLIGHT
85        CQXR    = CLIGHQ*XNC1R
86        NR      = NRMAXI
87        NZ      = NZMAXI
88        NRM1    = NR-1
89        NZM1    = NZ-1
90  C
91  C-------------------------------------------------------------
92  C  1.0      TEST IF THE LASER IS ON OR IF REZONING MESH.
93  C-------------------------------------------------------------
94  C
95  C  (A) IF TIME < SWITCH-ON-TIME, RETURN.
96  C  (B) IF TIME > SWITCH-OFF-TIME, SWITCH OFF LASER.
97  C  (C) IF ENERGY SUPPLIED > ENERGY AVAILABLE, SWITCH LASER OFF.
98  C  (D) IF NOT( A OR B OR C ), SWITCH LASER ON.
99  C  (E) IF CASTOR IS RESTARTED, SWITCH LASER ON.
100 C  (F) IF NOT ( A OR B OR C OR D OR E ) PROCEED.
101 C-------------------------------------------------------------
102 C
103 C  CASE ( A )
104 C
```

```
105         IF (TIME .LT. XLAS1(1))     RETURN
106   C
107   C  CASE ( B ) AND ( C )
108   C
109         IF (TIME .LE. XLAS1(2))      THEN
110           IF (ELAS1 .LE. XLAS1(10)) THEN
111   C
112   C  CASE ( D )
113   C
114               IF (XLAS1(9) .LT. 0.0)    GO TO 200
115   C
116   C  CASE ( E )
117   C
118         IF (NLRES)                      GO TO 200
119   C
120   C  CASE ( F )
121   C
122                                         GO TO 400
123           END IF
124         END IF
125   C
126   C-------------------------------------------------------------
127   C  BLOCK 1.
128   C             IF THE LASER SHOULD BE TURNED OFF, ZERO
129   C             ALL COEFFICIENTS, AND SWITCH IT OFF
130   C             BY SETTING XLAS1(9) = -1
131   C
132          DO 150 JR=1,NR
133             XPLAS1(JR)     = 0.
134          DO 151 JZ=1,NZ
135             ABSLAS(JR,JZ) = 0.
136             EMSR(JR,JZ)    = 0.
137             FLUX(JR,JZ)    = 0.
138             EMSZ(JR,JZ)    = 0.
139    151    CONTINUE
140    150    CONTINUE
141   C
142          MLASER    = .FALSE.
143          XLAS1(9) = -1.
144   C
145          RETURN
146   C
147   C  END OF BLOCK 1.
148   C-------------------------------------------------------------
149   C  BLOCK 2.    STEPS WHEN LASER IS FIRST TURNED ON.
150   C-------------------------------------------------------------
151   C
152    200  CONTINUE
153   C
154   C   2.1  SUBDIVIDE THE Z HYDRODYNAMIC MESH
155   C
156   C  DEFINE THE MAXIMUM NUMBER OF SUBDIVISIONS IN  Z
```

```
C  FROM CZ(JZ) TO CZ(JZ+1). THIS NUMBER IS ODD AND >= 1
C
      NDIVZ = MAX0(7,IFIX(XLAS1(15)+.5))
      IF (NDIVZ/2*2 .EQ. NDIVZ) NDIVZ = NDIVZ + 1
C
C     TEST WHETHER LASER IS ON. IF NOT, TURN IT ON.
C
      IF (ILASON) GO TO 400
      ILASON    = .TRUE.
      XLAS1(9) = 1.0
C
C-----------------------------------------------------------------
C  2.2   SELECT THE BEAM SHAPE USING ISHAPE. TWO SPATIAL BEAM
C        SHAPES ARE ALLOWED - A GAUSSIAN OR A PLANE WAVE
C        SUPERPOSED WITH A GAUSSIAN. THE LATTER ASSUMES THAT
C        THE GAUSSIAN INTENSITY IS NEGLIGIBLE AT THE OUTER
C        RADIAL BOUNDARY.
C
      ISHAPE = XLAS1(19) + 0.1
C
      IF (ISHAPE .GT. 3) THEN
          ISHAPE = 2
      ELSE
          ISHAPE = 1
      ENDIF
C
       IF (ISHAPE .EQ. 1) THEN
C
C  ISHAPE = 1 , GAUSSIAN INPUT BEAM IS SELECTED
C-----------------------------------------------------------------
C
C  2.3 . PARAMETERS FOR A GAUSSIAN INPUT BEAM.
C
C  ZSIG0  =  THE VACUUM SPOT SIZE AT THE FOCUS.
C  ZSIG1  =  THE SMALLEST SIGMA IN THE PLASMA.
C  ZFOC   =  FOCAL LENGTH
C  ZINC   =  INCOHERENCY PARAMETER
C  BRL1   =  THE WIDTH OF THE BEAM AT Z=CZ(1).
C  ZSIGZR = THE SIGMA USED TO SET UP THE LASER GRID.
C       INITIALLY, ZSIGZR = ZSIG0
C
      ZK     = 2. * FCPI/XL1
      ZFOC   = XLAS1(13)
      ZINC   = XLAS1(14)
      ZSIG0  = XLAS1(12)
      ZSIG1  = ZSIG0
      ZSIGZR = ZSIG0
      ZSIG0Q = ZSIG0 * ZSIG0
      ZSIG0R = 1.0 / ZSIG0Q
C
C  INITIAL BIGGEST SPOT SIZE AT INPUT END(Z=CZ(1)) IS BRL1
C
```

```
209         ZTEMP = ZINC*(ZFOC-CZ(1))/(ZK*ZSIG0Q)
210         BRL1  = ZSIG0*SQRT(1.0+ZTEMP*ZTEMP)
211   C
212          ELSE
213   C
214   C  ISHAPE = 2 , UNIFORM BEAM IS SELECTED.
215   C-----------------------------------------------------------------
216   C
217   C  2.4    PARAMETERS FOR A UNIFORM BEAM SUPERPOSED WITH
218   C         A GAUSSIAN PERTURBATION.
219   C
220   C  ZEPS   = A FRACTION OF THE INCIDENT PLANE WAVE INTENSITY.
221   C  ZSIG0  = SIGMA OF THE GAUSSIAN PERTURBATION AT THE BOUNDARY.
222   C  ZINC   = INCOHERENCY PARAMETER
223   C  ZSIGZR = THE SIGMA USED TO SET UP THE LASER GRID.
224   C  BRL1   = THE WIDTH OF THE GAUSSIAN BEAM AT Z=CZ(1)
225   C
226         ZINC   = XLAS1(14)
227         ZK     = 2.0*FCPI/XL1
228         ZEPS   = XLAS1(13)
229         ZSIG0  = XLAS1(12)
230         ZSIG1  = ZSIG0
231         ZSIGZR = ZSIG0
232         ZSIG0Q = ZSIG0 * ZSIG0
233         ZSIG0R = 1.0 / ZSIG0Q
234         BRL1   = ZSIG0
235   C
236         END IF
237   C
238   C-----------------------------------------------------------------
239   C  3.0   SET UP THE LASER MESH WHEN LASER IS FIRST TURNED ON
240   C-----------------------------------------------------------------
241   C
242   C  3.1  THE HYDRO MESH CR MESH CAN BE SUBDIVIDED INTO A FINER
243   C       LASER MESH ZR. THE RADIAL (FINE) MESH SPACING ZDR0 IS
244   C       XLAS1(7). IF XLAS1(7) = 0.0 THEN STOP.
245   C       ZINCR IS THE 1/INCOHERENCY PARAMETER(ZINC)
246   C
247         ZINCR = 1.0/ZINC
248         ZDR0  = XLAS1(7)
249         IF (ZDR0 .LE. 0.0 ) THEN
250   C-----------------------------------------------------------------
251         WRITE(6,305)
252    305  FORMAT('THE RADIAL LASER MESH DR IS ZERO, CHECK XLAS1(7)')
253         STOP
254         END IF
255   C
256   C  3.2  IREPR(JR) IS THE # OF LASER POINT INSIDE THE REGULAR
257   C       GRID FROM J TO J+1.
258   C       TEMP1 CONTAINS LASER MESH SPACING BETWEEN JR AND JR+1
259   C       CONVERTED IREPR TO ODD NUMBER FOR BOOK KEEPING PURPOSE.
260   C
```

```
261         DO 315 JR=1,NRM1
262          IREPR(JR)  = DRH(JR)/ZDR0 + 0.5
263          IF (IREPR(JR)/2*2 .EQ. IREPR(JR)) IREPR(JR)=IREPR(JR)+1
264          TEMP1(JR) = DRH(JR)/FLOAT(IREPR(JR))
265   315    CONTINUE
266   C
267   C  3.3  CALCULATE IMESH
268   C       IMESH IS THE TOTAL # OF LASER POINTS FROM R(0) TO R(N)
269   C
270         IBCR(1)     = 1
271          DO 320 JR=1,NRM1
272            IBCR(JR+1) = IBCR(JR) + IREPR(JR)
273   320    CONTINUE
274         IMESH       = IBCR(NR)
275         IMESH1      = IMESH + 1
276   C
277   C 3.5  EXTEND THE RADIAL MESH BY AN AMOUNT XLAS1(11) OUTSIDE
278   C      THE PLASMA. IMESHX IS THE TOTAL # OF LASER GRID POINTS.
279   C
280         NEXTND = XLAS1(11) + .5
281         NEXTND = MIN0(NRZMAX,NEXTND)
282         IMESHX = IMESH + NEXTND
283         IMSHX1 = IMESHX-1
284   C
285   C  CHECK WHETHER THE NUMBER OF LASER POINTS EXCEED MAXIMUM
286   C  DEFINED ARRAY DIMENSION.
287   C
288         IF (IMESHX .GT. NFMMAX) THEN
289         WRITE(6,345) IMESHX
290    345  FORMAT('FATAL ERROR IN HEATER: IMESHX=',I4,'IS TOO LARGE.')
291         STOP
292         END IF
293   C
294   C  JRS(I) IS THE LOWER JR VALUE ASSOCIATED WITH LASER POINT I
295   C  JRS(I) CORRESPONDS TO CURRENT HYDRO CELL JR AND USES TO
296   C  INTERPOLATE FROM HYDRO CONDITIONS
297   C
298        JRS(1)    = 1
299         DO 350 JR=1,NRM1
300           DO 351 JS=1,IREPR(JR)
301             JRS(IBCR(JR)+JS) = JR
302    351     CONTINUE
303    350   CONTINUE
304   C
305   C  FOR POINTS OUTSIDE CR(NR) ( OUTSIDE PLASMA )
306   C
307          DO 355 IB=IMESH1,IMESHX
308           JRS(IB) = NRM1
309    355   CONTINUE
310   C
311   C  JRP1S(I) IS THE UPPER JR ASSOCIATED WITH LASER POINT I
312   C  JRS1S CORRESPONDS TO NEXT HYDRO CELL JR + 1.
```

```
313  C
314          DO 356 IB=1,IMESHX
315           JRP1S(IB) = JRS(IB) + 1
316   356    CONTINUE
317  C
318  C  TEMPS IS NUMBER OF SMALL INTERVALS FROM LOWER JR TO
319  C  UPPER LASER POINT JR+1. ZR CORRESPONDS TO THE LASER RADIAL
320  C  POSITIONS FROM THE AXIS.
321  C
322          ZR(1)       = 0.
323          DO 360 IB=2,IMESH
324           TEMPS(IB) = FLOAT(IB-IBCR(JRS(IB)))
325           ZR(IB)    = CR(JRS(IB)) + TEMPS(IB)*TEMP1(JRS(IB))
326   360    CONTINUE
327  C
328  C  FOR POINTS OUTSIDE CR(NR)
329  C
330          DO 362 IBX=1,NEXTND
331           ZR(IBX+IMESH) = CR(NR) + FLOAT(IBX)*TEMP1(JRS(IMESH))
332   362    CONTINUE
333  C
334  C  CALCULATE THE LASER SPACING ZDR ( ZR(IB)-ZR(IB-1) ),
335  C  RECIPROCAL 1/ZR, ZR**2, RECIPROCAL 1/ZDR AND STORED IN
336  C  ARRAYS TO AVOID REPEAT CALCULATION.
337  C
338           ZRQS(1)  = ZR(1)**2
339          DO 364 IB=2,IMESHX
340           ZDR(IB)  = ZR(IB) - ZR(IB-1)
341          ZDRRS(IB) = 1.0/ZDR(IB)
342           ZRRS(IB) = 1.0/ZR(IB)
343           ZRQS(IB) = ZR(IB)**2
344   364    CONTINUE
345  C
346  C  INTERPOLATION FACTORS FOR USE ON LASER GRID. THESE FACTORS
347  C  ALLOW TO COMPUTE VALUES SUCH AS DENSITY, TEMPERATURE,
348  C  CHARGE NUMBER AND ABSORPTION COEFFICIENTS CORRESPONDS TO
349  C  EACH LASER GRID FROM THE KNOWN HYDRO MESH.
350  C  ZRHOS ARE FORWARD INTERPOLATION FACTORS (J -> J+1)
351  C  ZRHODS ARE BACKWARD INTERPOLATION FACTOR (J+1 -> J)
352  C
353           ZRHOS(1)  = 1.0
354          DO 370 IB=2,IMESH
355           ZRHOS(IB) = TEMPS(IB)/FLOAT(IREPR(JRS(IB)))
356   370    CONTINUE
357  C
358          DO 372  IB=IMESH1,IMESHX
359           ZRHOS(IB) = 1.0
360   372    CONTINUE
361  C
362          DO 374  IB=1,IMESHX
363           ZRHODS(IB) = 1.0 - ZRHOS(IB)
364   374    CONTINUE
```

```
C
C  JMINS AND JMAXS CORRESPOND TO THE POSITION OF HYDRO-CELL
C  FROM JR TO JR+1. THESE VALUES DEFINE THE LIMIT OF
C  INTEGRATION FROM JR TO JR+1.
C
         JMINS(1) = 1
       DO 378 JR=2,NRM1
         JMINS(JR) = IBCR(JR) - IREPR(JR-1)/2
 378   CONTINUE
       JMINS(NR) = IBCR(NRM1) + IREPR(NRM1)/2 + 1
C
       DO 380 JR=1,NRM1
         JMAXS(JR) = JMINS(JR+1)-1
 380   CONTINUE
         JMAXS(NR) = IMESHX
C
C  IRRS(IL) DEFINES JR OF HYDRO-CELL ASSOCIATED WITH
C  LASER POINT IL
C
       DO 382 JR=1,NR
         DO 383 IL=JMINS(JR),JMAXS(JR)
           IRRS(IL) = JR
 383     CONTINUE
 382   CONTINUE
C
C-----------------------------------------------------------------
C
C  CALL SETRDR TO EVALUATES INTEGRAL ARROUND MESH POINTS FORM
C  J=1 (R=0) TO IMESHX OF THE LASER GRID POINTS.
C  ZRDR(J)=INTEGRAL( ZR(J)*ZDR(J) ) FROM ( ZR(J-1)+ZR(J) )/2
C  TO ( ZR(J)+ZR(J+1) )/2
C
       CALL SETRDR(ZR,ZRDR,IMESHX)
C
C-----------------------------------------------------------------
C
C  3.7   SET UP ZT1,ZT2 MATRICES FOR CUBIC SPLINE METHOD
C        FIRST CALCULATE ZALFA,ZBETA,ZGAMA,ETC...
C        ( SEE FLECK'S PAPER FOR DETAIL CALCULATION OF
C        EACH COEFFICIENTS )
C
       ZALFA(2) = 0.
       ZBETA(2) = 0.
       ZGAMA(2) = 0.
C
       DO 385 JL = 3,IMESHX
       ZGAMA(JL) = 1./ALOG(ZR(JL)*ZRRS(JL-1))
       ZALFA(JL) = ZGAMA(JL)*(ZRQS(JL)*(ZR(JL-1)/4.-ZR(JL)/9.)
      &          - 5./36.*ZR(JL-1)*ZRQS(JL-1))*ZDRRS(JL)
       ZBETA(JL) = ZGAMA(JL)*(ZRQS(JL-1)*(ZR(JL)/4.-ZR(JL-1)/9.)
      &          - 5./36.*ZR(JL)*ZRQS(JL))*ZDRRS(JL)
 385   CONTINUE
```

```
417   C
418   C  NOW EVALUATE T1 AND T2 MATRICES BY CALCULATING LOWER,
419   C  UPPER AND DIAGONAL COEFFICIENTS FOR T-VECTORS.
420   C
421         ZT1(1,1) = 0.0
422         ZT2(1,1) = 0.0
423         ZT1(1,2) = -5./36.*ZRQS(2)
424         ZT2(1,2) = 1.0
425         ZT1(1,3) = 0.8*ZT1(1,2)
426         ZT2(1,3) = -1.0
427   C
428         DO 390 JL = 2,IMSHX1
429         ZT1(JL,1) = -ZBETA(JL)/ZR(JL)-ZRQS(JL)/(6.*ZDR(JL))
430         ZT1(JL,2) = ZBETA(JL+1)/ZR(JL)+ZR(JL)*(ZR(JL+1)/2.
431        1           - ZR(JL)/3.)*ZDRRS(JL+1)-ZALFA(JL)/ZR(JL)
432        2           - ZR(JL)*(ZR(JL)/3.-ZR(JL-1)/2.)*ZDRRS(JL)
433         ZT1(JL,3) =  ZALFA(JL+1)/ZR(JL)-ZRQS(JL)/(6.*ZDR(JL+1))
434         ZT2(JL,1) = -ZGAMA(JL)/ZR(JL)
435         ZT2(JL,3) = -ZGAMA(JL+1)/ZR(JL)
436         ZT2(JL,2) = -ZT2(JL,1)-ZT2(JL,3)
437    390  CONTINUE
438   C
439   C  USE FLEX CONDITIONS AT THE BOUNDARY:
440   C  R=0 & R=RMAX, I.E. M=0.
441   C
442         ZT2(IMESHX,1) = 0.
443         ZT2(IMESHX,2) = 0.
444         ZT2(IMESHX,3) = 0.
445         ZT1(IMESHX,1) = 0.
446         ZT1(IMESHX,2) = 1.
447         ZT1(IMESHX,3) = 0.
448   C
449   C  COPY REAL T1 TO COMPLEX T1 FOR PASSING TO CYCLIC REDUCTION
450   C  S/R. THIS IS USED TO CALCULATE LAPLACIAN ZM FOLLOWING THE
451   C  RELATION:
452   C           ZT1*ZM = ZT2*ZCFLUX
453   C  NOTE THAT THIS IS ONLY DONE ONCE UNLESS THE MESH IS REZONED
454   C
455         DO 492 JL = 1, IMESHX
456           ZCT1(JL,1) = ZT1(JL,1) * (1.,0.)
457           ZCT1(JL,2) = ZT1(JL,2) * (1.,0.)
458           ZCT1(JL,3) = ZT1(JL,3) * (1.,0.)
459    492 CONTINUE
460   C
461   C-----------------------------------------------------------
462   C  DEFINE REFRAC, THE WEIGHTING FACTOR USED TO ATTENUATE
463   C  DIFFRACTION AT THE EDGE OF PLASMA COLUMN. THIS FACTOR
464   C  DEPENDS ON THE NUMBER BY WHICH IMESHX IS DIFFERENT FROM
465   C  IMESH. IT ALSO AFFECTS THE DEFINITION OF WRFRAC WHICH
466   C  APPEARS LATER.                        R. MARCHAND  SEP. 1985
467   C
468   C  IF THE LASER REACHES THE PLASMA BOUNDARY A WEIGHT FUNCTION
```

```
469  C  2*(X-A)**3-1.5*(X-A)+0.5 IS USED TO ELIMINATE REFLECTION
470  C  FROM THE BOUNDARY.
471  C  NOW ATTENUATE THE ZT2 MATRIX OUTSIDE THE PLASMA
472  C  GRID TO REDUCE REFLECTION.
473  C
474  C    IF XLAS1(15) = 1 DO NOT ATTENUATE THE T2 MATRIX
475  C    IF XLAS1(15) .NE. 1 USE WEIGHTING FACTOR TO ATTENUATE T2
476  C
477         IF( NEXTND .GT. 0 .AND. XLAS1(15) .NE. 1.0) THEN
478          REFRAC     = 1./FLOAT(NEXTND)
479         DO 395 JL=IMESH1,IMESHX
480           WRFRAC    = REFRAC*(JL-IMESH)-0.5
481           WRFRAC    = (2.*WRFRAC*WRFRAC-1.5)*WRFRAC+.5
482           ZT2(JL,1) = WRFRAC*ZT2(JL,1)
483           ZT2(JL,2) = WRFRAC*ZT2(JL,2)
484           ZT2(JL,3) = WRFRAC*ZT2(JL,3)
485   395   CONTINUE
486         END IF
487  C
488  C  END OF BLOCK 2
489  C
490  C-----------------------------------------------------------------
491  C  BLOCK 3       INPUT LASER POWER AT CURRENT TIME STEP.
492  C                (IE. INPUT POWER AT THE ENTRANCE PLASMA).
493  C-----------------------------------------------------------------
494  C
495  C   IF THE LASER IS ON EXECUTION BEGINS HERE.
496  C   SELECT DIFFERENT KIND OF PULSES IKIND AND
497  C   CALCULATE THE INPUT POWER AT THE ENTRANCE PLASMA ZPL1
498  C
499   400  IKIND = XLAS1(8) + 0.0001
500  C
501         IF (IKIND .EQ. 1) THEN
502  C
503  C  4.1           COMPRESSION PULSE
504  C
505         Z4   = 1.0
506         IF(XLAS1(2).NE.0.0) ZA1 = 1.0 - TIME/XLAS1(2)
507         IF(XLAS1(3).NE.0.0) Z4  = XLAS1(3)
508         ZPL1 = XLAS1(4) * ZA1**Z4
509  C
510         ELSEIF (IKIND .EQ. 2) THEN
511  C
512  C  4.2           GAUSSIAN PULSE
513  C
514         ZMIN = EXPMIN
515         ZMAX = EXPMAX
516         ZA1  = (TIME - XLAS1(3))/XLAS1(4)
517         ZA2  = 0.5 * ZA1 * ZA1
518         ZA2  = AMAX1(ZMIN,AMIN1(ZMAX,ZA2))
519         ZPL1 = XLAS1(5) * EXP(-ZA2)
520  C
```

```
      ELSEIF (IKIND .EQ. 3) THEN
C
C  4.3          TRIPLE RAMP PULSE
C
      IF (TIME.GT.XLAS1(3)) GO TO 432
      ZPL1 = XLAS1(5)*TIME/XLAS1(3)
      GO TO 500
C
 432  IF(TIME.GT.XLAS1(4)) GO TO 434
      ZPL1 = (XLAS1(5)*(XLAS1(4)-TIME)+XLAS1(6)*(TIME-XLAS1(3)))
     &     / (XLAS1(4)-XLAS1(3))
      GO TO 500
C
 434  ZPL1 = XLAS1(6)*(XLAS1(2)-TIME)/(XLAS1(2)-XLAS1(4))
C
      ELSEIF (IKIND .EQ. 4) THEN
C
C  4.4     PLANE WAVE WITH GAUSSIAN PERTURBATION.
C          XLAS1(5) CONTAINS THE PLANE WAVE INTENSITY.
C
      CONTINUE
      ZPL1 = FCPI*XLAS1(5)*( CR(NR)*CR(NR)
     &     + ZEPS*ZSIG0Q )
C
 500  ENDIF
C
       DTNEW = DT
       PNEW  = ZPL1
      CALL SHORT(ISHORT,DTNEW,PNEW,ACUMT)
      IF (ISHORT .EQ. 1 ) RETURN
      ACCUMT = 0.0
C
C   NORMALIZED INPUT POWER ( ZPL1/PI ).
C
      ZNORM = ZPL1/FCPI
      PWRIN = ZPL1
C
C  END OF BLOCK 3
C-----------------------------------------------------------------
C  BLOCK 4.
C
C  CALCULATE ABSORPTION COEFFICIENTS ZABS ON CR-CZ MESH
C -----------------------------------------------------------------
C
C  5.1     SET NUMERICAL CONSTANT
C
      ZA1 = 13.51/(XL1Q)
C
C  5.2     MAIN LOOP THRU R,Z
C
C       CALCULATE Z*(SPITZER COULOMB LOGARITHM)
C       CALCULATE ABSORPTION COEFFICIENTS ZABS
```

```
C
C
      DO 550 JZ=1,NZ
      DO 551 JR=1,NR
      ZX = AMIN1(AMAX1(ALOG10(XL1*AZ(JR,JZ)**2)+4.5,-2.5),2.5)
      ZY = AMIN1(AMAX1(ALOG10( TE(JR,JZ)
     &   / (11605.*AZ(JR,JZ)**2 ))-2.,-2.),2.)
      ZXZ = AMIN1(0.9999,XNE(JR,JZ)*XNC1R)
      ZZLOGL = AZ(JR,JZ)*((0.126149*ZX
     1       + 2.230034)*ZX+8.007157
     2       + ZY*((0.314866-0.078475*ZX)*ZX+2.703057
     3       + ZY*((0.006118*ZX-0.120930)*ZX+0.103552)))
      ZNENC  = ZXZ*ZXZ/SQRT(1.-ZXZ)
      ZABS(JR,JZ) = ZA1*ZZLOGL*ZNENC/TE(JR,JZ)**1.5
  551 CONTINUE
  550 CONTINUE
C
C   CLEAR ALL COEFFICIENTS EMSR, EMSZ, ABSLAS, FLUX
C
       DO 560 JR=1,NR
          XPLAS1(JR)     = 0.
       DO 570 JZ=1,NZ
          ABSLAS(JR,JZ) = 0.
          EMSR(JR,JZ)    = 0.
          FLUX(JR,JZ)    = 0.
          EMSZ(JR,JZ)    = 0.
  570   CONTINUE
  560   CONTINUE
C
C  END OF BLOCK 4.
C-----------------------------------------------------------------
C  BLOCK 5
C
C   EVALUATE FLUXES  AND LAPLACIAN AT THE ENTRANCE PLASMA
C-----------------------------------------------------------------
C
C   SELECT GAUSSIAN BEAM IF ISHAPE = 1
C
      IF (ISHAPE .EQ. 1) THEN
C
C  6.1   GAUSSIAN CROSS-SECTION AT BOUNDARY, Z=CZ(1)
C
C   ZNE IS THE ELECTRON DENSITY ON THE EDGE Z=CONSTANT=CZ(1)
C   ZK0 IS THE LASER WAVENUMBER AT A DENSITY ZNE.
C   ZSIGZ2 = BEAM WIDTH SQUARED AT Z=CONSTANT=CZ(1)
C   ZR4SIG IS INITIALLY FOUR TIMES THE BEAM WIDTH AT Z=CZ(1)
C   THE FLUX IS CALCULATED OUT TO ZR4SIG ON THE LASER MESH.
C   ZNE IS NEVER ALLOWED TO EXCEED 0.99 * CRITICAL
C
C   FIRST INITIALIZE THE ZCQ AND FLUX ARRAY TO ZERO
C   ON LASER GRID AT Z=CZ(1) TO ERASE PRIOR CALLS TO S/R
C
```

```
      DO 601 JL=1,IMESHX
      ZFLUX(JL)  = 0.0
      ZCQ(IL)    = (0.,0.)
      ZCFLUX(JL) = (0.,0.)
      ZM(JL)     = (0.,0.)
601   CONTINUE
      ZNE    = AMIN1(XNE(1,1),XNC1F)
      ZK0    = ZK*SQRT(1.-ZNE*XNC1R)
      ZCFAC  = CMPLX(1.,-ZTEMP)
      ZSIGZ2 = BRL1 * BRL1
      ZDS2DZ = -2.*ZTEMP*ZINC + 1.E-30
      ZR4SIG = 4.*BRL1
      ZNFAC  = SQRT(ZPL1/(FCPI*ZSIG0Q))/ZCFAC
C
C   INITIALLY THE GRID POINT AT Z=ZR4SIG IS I4SIG=IMESHX
C
      I4SIG = IMESHX
C
C-----------------------------------------------------------------
C   EVALUATE COMPLEX FLUX (ZCFLUX) AND FLUX (ZFLUX)
C   ON BOUNDARY Z=CZ(1)  AS A FUNCTION OF R
C   ZM IS THE LAPLACIAN 1/R * D/DR (R * D/DR(ZCFLUX) )
C
      DO 610 JL=1,IMESHX
      ZCEXP      = -.5*ZRQS(JL)*ZSIG0R/ZCFAC
      ZCFLUX(JL) = ZNFAC*CEXP(ZCEXP)
      ZFLUX(JL)  = CABS(ZCFLUX(JL))**2
C
      ZM(JL) = 2.*( -ZCEXP-(1.,0.) )*ZCFLUX(JL)*ZSIG0R/ZCFAC
610   CONTINUE
C
C   DEFINE INDEX ON LASER MESH AT WHICH ZR(JL)=4.*BEAM WIDTH
C
      DO 611 JL=1,IMESHX
      I4SIG = JL
      IF( ZR(JL) .GE. ZR4SIG ) GO TO 612
611   CONTINUE
612   I4SGM1 = I4SIG - 1
      I4SGP1 = I4SIG + 1
C
      ELSE
C-----------------------------------------------------------------
C
C   6.2 PLANE WAVE + GAUSSIAN PERTURBATION AT CZ(1).
C
C   ZNE IS THE ELECTRON DENSITY AT THE EDGE Z = CZ(1).
C   ZK0 IS THE LASER WAVENUMBER AT Z = CZ(1).
C   ZSIGZ2 = SQUARE OF THE BEAM WIDTH AT Z = CZ(1).
C   ZDS2DZ = ZSIGZ2
C   ZR4SIG IS INITIALLY SET TO CR(NR).
C   ZPWB IS THE INCIDENT PLANE WAVE INTENSITY.
C
```

```
677         ZNE    = AMIN1(XNE(1,1),XNC1F)
678         ZK0    = ZK * SQRT(1. - ZNE*XNC1R)
679         ZSIGZ2 = ZSIG0Q
680         ZDS2DZ = ZSIGZ2
681         ZR4SIG = CR(NR)
682         ZPWB   = XLAS1(5)
683   C
684   C     ADD RIPPLE INTENSITY TO THE PLANE WAVE FLUX ON THE Z
685   C     BOUNDARY.
686   C     TEMPS IS THE MAXIMUN EXPONENTIAL ALLOWABLE VALUE BETWEEN
687   C     EXPMIN AND THE  ZR**2/SIGMA**2 TO PREVENT UNDERFLOW
688   C
689         DO 699 JL = 1,IMESHX
690          TEMPS(JL)   = EXP(AMAX1(EXPMIN,-ZRQS(JL)/ZSIGZ2))
691          ZFLUX(JL)   = ZPWB + ZEPS * ZPWB * TEMPS(JL)
692          ZCFLUX(JL) = SQRT(ZFLUX(JL))
693    699   CONTINUE
694   C
695   C     EVALUATE THE LAPLACIAN ON THE BOUNDARY Z=CZ(1)
696   C
697         ZSCALE = SQRT(ZPWB)/(ZSIGZ2)*ZEPS
698         DO 603 JL = 1,IMESHX
699                ZMFAC  = 1.0 + ZEPS * TEMPS(JL)
700                ZM(JL) = ( -2.0 + (ZRQS(JL)/ZSIGZ2)
701        1                * (2.0 - ZEPS*TEMPS(JL)/ZMFAC) )
702        2                * ZSCALE/SQRT(ZMFAC)*TEMPS(JL)
703    603   CONTINUE
704         I4SIG  = IMESHX
705         I4SGM1 = I4SIG - 1
706         I4SGP1 = I4SIG + 1
707   C
708         END IF
709   C
710   C  6.3    RESCALE THE FLUX ON THE LASER MESH AT Z=CZ(1) SO THAT
711   C         THE EXACT INPUT POWER IS OBTAINED.
712   C
713         TGR = 0.0
714         DO 604 J=1,IMESHX
715    604    TGR         = TGR + ZRDR(J)*ZFLUX(J)
716           ZZ          = ZPL1/(ZTWOPI*TGR)
717         DO  625 JL = 1,I4SIG
718           ZCFLUX(JL) = SQRT(ZZ)*ZCFLUX(JL)
719           ZFLUX(JL)  = ZZ*ZFLUX(JL)
720    625   CONTINUE
721   C
722   C-----------------------------------------------------------------
723   C  6.4    COMPUTE DENSITY NES, ABSORPTION ABSS, TEMPERATURE
724   C         TES AND CHARGE AZS ON LASER GRID AT Z = CZ(1).
725   C
726         ZNES(1)  = XNE(1,1)
727         ZTES(1)  = TE(1,1)
728         ZAZS(1)  = AZ(1,1)
```

```
729         ZABSS(1) = ZABS(1,1)
730   C
731         DO 630 JL=2,IMESHX
732           ZNES(JL)  = ZRHODS(JL) * XNE(JRS(JL),1)
733        &            + ZRHOS(JL)  * XNE(JRP1S(JL),1)
734           ZTES(JL)  = ZRHODS(JL) * TE(JRS(JL),1)
735        &            + ZRHOS(JL)  * TE(JRP1S(JL),1)
736           ZAZS(JL)  = ZRHODS(JL) * AZ(JRS(JL),1)
737        &            + ZRHOS(JL)  * AZ(JRP1S(JL),1)
738           ZABSS(JL) = ZRHODS(JL) * ZABS(JRS(JL),1)
739        &            + ZRHOS(JL)  * ZABS(JRP1S(JL),1)
740    630  CONTINUE
741   C
742   C    COMPUTE THE LANGDON CORRECTION FACTOR AT Z=CZ(1)
743   C
744         DO 640 JL=1,IMESHX
745           ZNE0S(JL)  = AMIN1(ZNES(JL),XNC1F)
746           ZVG0S(JL)  = SQRT(1.-ZNE0S(JL)*XNC1R)
747           ZARG       = (XL1QF*ZAZS(JL)*ZFLUX(JL)/(ZVG0S(JL)
748        &               * ZTES(JL)))**.75
749           WLANGD(JL) = 1.0-0.553*ZARG/(0.375+ZARG)
750    640  CONTINUE
751   C
752   C 6.5     INTEGRATE FLUX ON LASER MESH AND RESCALE XPLAS1 USED
753   C         ON PLASMA GRID TO DEFINE THE LASER FLUX ACCURATELY.
754   C
755         DO 665 JR=1,NR
756         TGR = 0.0
757         DO 666 J=JMINS(JR),JMAXS(JR)
758    666    TGR         = TGR + ZRDR(J)*ZFLUX(J)
759           XPLAS1(JR) = TGR/RDR(JR)
760           FLUX(JR,1) = XPLAS1(JR)
761    665  CONTINUE
762   C
763   C 6.6     PREPARE THE ZCQOLD ARRAY WHICH IS THE Q ARRAYS AT
764   C         THE ENTRANCE PLASMA.
765   C         FIRST, CALCULATE Q ARRAY ON THE AXIS
766   C
767         ZK0        = ZK*SQRT(1.-ZNE0S(1)*XNC1R)
768         ZK0OLD     = ZK0
769         ZABS0      = ZABSS(1)*WLANGD(1)
770         ZCQOLD(1)  = CMPLX(0.,-ZABS0)
771         ZK0Q       = ZK0*ZK0
772         ZK0R       = 1.0/ZK0
773   C
774   C     LOOP THROUGH FINE MESH FROM 2 -> IMESHX
775   C
776         DO 700 IB=2,IMESHX
777           ZKRS(IB)    = ZK*SQRT(1.-ZNE0S(IB)*XNC1R)
778           ZQREAS(IB)  = (ZK0Q-ZKRS(IB)**2)*ZK0R
779           ZQIMAS(IB)  = -ZABSS(IB)*WLANGD(IB)
780           ZCQOLD(IB)  = CMPLX(ZQREAS(IB)*ZINCR,ZQIMAS(IB))
```

```
  700  CONTINUE
         ZPOND = 0.0
         ZPABS = 0.0
C
C  END OF BLOCK 5
C
C-----------------------------------------------------------------
C  BLOCK 6      CALCULATE NEW AMPLITUDE AT ALL R,Z
C-----------------------------------------------------------------
C
C ****************************************************************
C *  SOLUTION OF FLUX AND ABSORBED POWER AT ALL R AND Z.         *
C *  CALCULATE RADIAL AND AXIAL PONDEROMOTIVE FORCES             *
C *  CALCULATE WORK DONE AND SCALING THE BEAM TO ENSURE          *
C *  CONSERVATION OF ENERGY                                      *
C ****************************************************************
C
C-----------------------------------------------------------------
C -- START OUTER Z LOOP FROM JZ=1 TO JZ=NZ-1 ON PLASMA GRID --
C-----------------------------------------------------------------
C
       DO 710 JZ=1,NZM1
       JZP1  = JZ+1
C
C   7.1           PICK STEP SIZE, REPETITION FACTOR
C
       ZDZC  = CZ(JZ+1)-CZ(JZ)
       ZDZCR = 1/ZDZC
       ZDZL  = ZDZC
C
C  CALCULATE ABSORPTION STEP SIZE BY IMPOSING CONDITIONS, IN
C  WHICH NO MORE THAN 10% OF LASER IS ABSORBED IN ONE STEP
C  AND THE BEAM WIDTH CHANGES LESS THAN 5% OF IT PREVIOUS STEP.
C
       DO 712 IB=2,I4SIG
         ZKA1S(IB) = (ZRHOS(IB)*ZABS(JRP1S(IB),JZ)
      &            + ZRHODS(IB)*ZABS(JRS(IB),JZ))*WLANGD(IB)
         ZKA2S     = (ZRHOS(IB)*ZABS(JRP1S(IB),JZP1)
      &            + ZRHODS(IB)*ZABS(JRS(IB),JZP1))*WLANGD(IB)
         ZA1S(IB)  = (ZKA2S-ZKA1S(IB))/(ZKA1S(IB)*ZDZC)
  712  CONTINUE
C
       DO 714 IB=2,I4SIG
          IF (ZA1S(IB).GE.0.05*ZKA1S(IB)) THEN
            ZDZ0 = (SQRT(1.+0.2*ZA1S(IB)/ZKA1S(IB))-1.)/ZA1S(IB)
          ELSE
            IF (ABS(ZA1S(IB)).LE.0.05*ZKA1S(IB)) THEN
              ZDZ0 = 0.1 / ZKA1S(IB)
            ELSE
             IF (ZA1S(IB).LE.- 5.*ZKA1S(IB)) GO TO 714
             ZDZ0 = -(SQRT(1.+0.2*ZA1S(IB)/ZKA1S(IB))+1.)/ZA1S(IB)
            END IF
```

```
833             END IF
834             ZDZL = AMIN1(ZDZL,ZDZ0)
835   714     CONTINUE
836   C
837           ZDZ0 = 0.05 * ABS(ZSIGZ2/ZDS2DZ)
838           ZDZL = AMIN1(ZDZL,ZDZ0)
839   C
840   C    IREPZ IS NUMBER OF LASER GRID POINTS FROM JZ TO JZ+1
841   C
842           IREPZ = ZDZC / ZDZL + 0.0001
843           IREPZ = (IREPZ/2*2) + 1
844           IREPZ = MIN0(IREPZ,NDIVZ)
845   C
846           ZDZL   = ZDZC / IREPZ
847           IREPZ1 = IREPZ + 1
848   C
849   C ---------- START OF INNER  LASER GRID Z-LOOP --------------
850   C     (BETWEEN CZ(JZ) AND CZ(JZ+1)),
851   C     I.E. FROM JIN=1 TO IREPZ+1
852   C ------------------------------------------------------------
853           DO 720 JIN = 1, IREPZ1
854   C
855   C     ZETA AND ZETAD ARE THE INTERPOLATION FACTORS WHERE ZETA
856   C     IS FORWARD AND ZETAD IS BACKWARD INTERPOLATION FACTORS
857   C
858             ZETA  = FLOAT(JIN - 1) / FLOAT(IREPZ)
859             ZETAD = 1. - ZETA
860   C
861   C     IF JIN = 1 => NO NEED TO COMPUTE NEW SOLUTION.
862   C     GO TO 750 TO CALCULUATE PONDEROMOTIVE FORCES,
863   C     LASER ABSORPTION ETC. AND SCALING BEAM SHAPE.
864   C
865             IF (JIN .EQ. 1) GO TO 750
866   C
867   C     CALCULATE DENSITY, TEMPERATURE, CHARGE NUMBER
868   C     ABSORPTION COEFFICIENT AT CURRENT LASER Z PLANE
869   C     BETWEEN CZ(JZ) AND CZ(JZ+1).
870   C
871           DO 723 IB = 1, IMESHX
872             ZNES(IB)  = ZRHODS(IB)*(ZETAD*XNE(JRS(IB),JZ)
873         1                         + ZETA*XNE(JRS(IB),JZP1))
874         2             + ZRHOS(IB)*(ZETAD*XNE(JRS(IB)+1,JZ)
875         3                         + ZETA*XNE(JRS(IB)+1,JZP1))
876             ZTES(IB)  = ZRHODS(IB)*(ZETAD*TE(JRS(IB),JZ)
877         1                         + ZETA*TE(JRS(IB),JZP1))
878         2             + ZRHOS(IB)*(ZETAD*TE(JRS(IB)+1,JZ)
879         3                         + ZETA*TE(JRS(IB)+1,JZP1))
880             ZAZS(IB)  = ZRHODS(IB)*(ZETAD*AZ(JRS(IB),JZ)
881         1                         + ZETA*AZ(JRS(IB),JZP1))
882         2             + ZRHOS(IB)*(ZETAD*AZ(JRS(IB)+1,JZ)
883         3                         + ZETA*AZ(JRS(IB)+1,JZP1))
884             ZABSS(IB) = (ZRHODS(IB)*(ZETAD*ZABS(JRS(IB),JZ)
```

```
885         1                          + ZETA*ZABS(JRS(IB),JZP1))
886         2               + ZRHOS(IB)*(ZETAD*ZABS(JRS(IB)+1,JZ)
887         3               + ZETA*ZABS(JRS(IB)+1,JZP1)))*WLANGD(IB)
888   723   CONTINUE
889   C
890         DO 725 IB = 1, IMESHX
891           ZNEOS(IB) = AMIN1(ZNES(IB),XNC1F)
892           ZVGOS(IB) = SQRT(1.-ZNEOS(IB)*XNC1R)
893           ZKRS(IB)  = ZK*ZVGOS(IB)
894   725   CONTINUE
895   C
896   C  7.2     CALCULATE Q(R)-ARRAY ZCQ AT CURRENT LASER Z PLANE
897   C           BETWEEN CZ(JZ) AND CZ(JZ+1).
898   C           ZCQQOLD(R) IS Q ARRAY AT PREVIOUS Z PLANE.
899   C
900         ZK0  = ZKRS(1)
901         ZK0Q = ZK0*ZK0
902         ZK0R = 1.0/ZK0
903         DO 730 IB=1,I4SIG
904           ZQREAS(IB) = (ZK0Q-ZKRS(IB)**2)*ZK0R
905           ZCQ(IB)    = CMPLX(ZQREAS(IB)*ZINCR,-ZABSS(IB))
906   730   CONTINUE
907         ZCK    = CMPLX(0.0,-4./ZDZL)
908         ZIKR   = ZINC/ZK0
909         ZIKRO  = ZINC/ZK0OLD
910   C
911   C    MODIFIED BY DANG T. OCT. 1990
912   C
913   C    CZKMQ IS THE COMPLEX VECTOR K - QOLD
914   C    CZKPQ IS THE COMPLEX VECTOR K + QNEW
915   C    CZT3  IS THE COMPLEX VECTOR T2 - T1*(K+QNEW)
916   C    CZE   IS THE COMPLEX VECTOR [-T2 - T1*(K-QOLD)]*ZCFLUX
917   C
918   C    CZT3 ARE TRIDIAGONAL MATRICES
919   C
920   C    SWEEP THROUGH THE RADIAL POINT FROM R(1) TO R(IMESHX)
921   C
922         DO 735 JL=1,I4SIG
923         CZKMQ(JL) = ZCK - ZCQOLD(JL)
924         CZKPQ(JL) = ZCK + ZCQ(JL)
925   735   CONTINUE
926   C
927   C    EVALUATE THE CZT3 COEFFICIENTS MATRICES
928   C    CZT3(JL,1) IS LOWER DIAGONAL ELEMENTS
929   C    CZT3(JL,2) IS DIAGONAL       ELEMENTS
930   C    CZT3(JL,3) IS UPPER DIAGONAL ELEMENTS
931   C
932         CZT3(1,1) = (0.0,0.0)
933         CZT3(1,2) = ZIKR*ZT2(1,2) - ZT1(1,2)*CZKPQ(1)
934         CZT3(1,3) = ZIKR*ZT2(1,3) - ZT1(1,3)*CZKPQ(2)
935   C
936         DO 737 JL=2,I4SGM1
```

```
937          CZT3(JL,1) = ZIKR*ZT2(JL,1) - ZT1(JL,1)*CZKPQ(JL-1)
938          CZT3(JL,2) = ZIKR*ZT2(JL,2) - ZT1(JL,2)*CZKPQ(JL)
939          CZT3(JL,3) = ZIKR*ZT2(JL,3) - ZT1(JL,3)*CZKPQ(JL+1)
940    737   CONTINUE
941   C
942          CZT3(I4SIG,1) = ZIKR*ZT2(I4SIG,1) - ZT1(I4SIG,1)
943         &                * CZKPQ(I4SGM1)
944          CZT3(I4SIG,2) = ZIKR*ZT2(I4SIG,2) - ZT1(I4SIG,2)
945         &                * CZKPQ(I4SIG)
946          CZT3(I4SIG,3) = (0.0,0.0)
947   C
948   C    CZE ARE RIGHT HAND SIDE VECTOR
949   C    VARIABLE CZE IS THE T4*F
950   C    CZE  IS THE COMPLEX VECTOR [-T2 - T1*(K-QOLD)]*ZCFLUX
951   C
952          CZE(1) = (-ZIKRO*ZT2(1,2) - ZT1(1,2)*CZKMQ(1))*ZCFLUX(1)
953         &       + (-ZIKRO*ZT2(1,3) - ZT1(1,3)*CZKMQ(2))*ZCFLUX(2)
954   C
955          DO 740 JL=2,I4SGM1
956    740   CZE(JL) = (-ZIKRO*ZT2(JL,1)-ZT1(JL,1)*CZKMQ(JL-1))
957         1        * ZCFLUX(JL-1) + (-ZIKRO*ZT2(JL,2) - ZT1(JL,2)
958         2        * CZKMQ(JL))*ZCFLUX(JL) + (-ZIKRO*ZT2(JL,3)
959         3        - ZT1(JL,3)*CZKMQ(JL+1))*ZCFLUX(JL+1)
960   C
961          CZE(I4SIG) = ( - ZIKRO*ZT2(I4SIG,1) - ZT1(I4SIG,1)
962         1           * CZKMQ(I4SGM1) ) * ZCFLUX(I4SGM1) +
963         2           ( - ZIKRO*ZT2(I4SIG,2) - ZT1(I4SIG,2)
964         3           * CZKMQ(I4SIG) ) * ZCFLUX(I4SIG)
965   C
966   C    COMPLETE SET UP OF THE TRIDIAGONAL SYSTEMS T3*F=T4*F
967   C    CALL CYCLIC REDUCTION TO FIND THE SOLUTION FOR ZCFLUX.
968   C
969          CALL CYCLIC(I4SIG,CZT3,CZE,ZCFLUX)
970   C
971   C-------------------------------------------------------------
972   C    CALCULATE THE REAL FLUX MAGNITUDE
973   C    NORMALIZING VECTOR ELEMENTS
974   C
975          DO 742 IL=1,I4SIG
976            ZFLUX(IL) = CABS(ZCFLUX(IL))**2
977    742   CONTINUE
978   C
979          ZNORM  = (0.25*ZRQS(2))*ZFLUX(1)
980          DO 743 IL = 2,I4SGM1
981           ZNORM = ZNORM + 0.25*(ZR(IL+1)-ZR(IL-1))*ZFLUX(IL)
982         &       * (ZR(IL+1)+2.*ZR(IL)+ZR(IL-1))
983    743   CONTINUE
984          ZNORM  = ZNORM + 0.25*(ZR(I4SIG)-ZR(I4SGM1))
985         &       * (3.*ZR(I4SIG)+ZR(I4SGM1))*ZFLUX(I4SIG)
986   C
987   C-------------------------------------------------------------
988   C    RECORD THE ZCQOLD ARRAY AND ERASE THE ZCQ ARRAY
```

```
  989  C
  990           ZK0OLD     = ZK0
  991          DO 746 IL = 1,IMESHX
  992           ZCQOLD(IL) = ZCQ(IL)
  993           ZCQ(IL)    = (0.,0.)
  994   746    CONTINUE
  995  C
  996  C   CALCULATE THE LAPLACIAN FOR THE NEXT STEP
  997  C   USING RELATION T1*M = T2*ZCFLUX
  998  C   T1 AND T2 ARE TRIDIAGONAL MATRICES
  999  C   CALL CYCLIC TO FIND THE SOLUTION FOR ZM
1,000  C   FIRST SET UP THE VECTOR D (RIGHT HAND SIDE)
1,001  C   PERFORM MATRIX MULTIPLICATION ZT2*ZCFLUX
1,002  C
1,003          CZE(1)  = (ZT2(1,2)*ZCFLUX(1) + ZT2(1,3)*ZCFLUX(2))
1,004         DO 748 JL=2,I4SGM1
1,005          CZE(JL) = (ZT2(JL,1)*ZCFLUX(JL-1) + ZT2(JL,2)
1,006        &          * ZCFLUX(JL) + ZT2(JL,3)*ZCFLUX(JL+1))
1,007   748   CONTINUE
1,008          CZE(I4SIG) = (ZT2(I4SIG,1)*ZCFLUX(I4SGM1)
1,009        &            + ZT2(I4SIG,2)*ZCFLUX(I4SIG))
1,010  C
1,011  C   ZCT1(JL,1)  ( LOWER DIAGONAL COEFF.)
1,012  C   ZCT1(JL,2)  ( DIAGONAL COEFF.)
1,013  C   ZCT1(JL,3)  ( UPPER DIAGONAL COEFF.)
1,014  C
1,015          CALL CYCLIC(I4SIG,ZCT1,CZE,ZM)   \
1,016  C
1,017   750   CONTINUE
1,018  C
1,019  C-------------------------------------------------------------
1,020  C
1,021  C  7.6  ABSORPTION ENERGY, PONDEROMOTIVE, BEAM SCALING TERMS.
1,022  C
1,023  C-------------------------------------------------------------
1,024  C   FIRST OF ALL DEFINE CONSTANTS SUCH AS GROUP VELOCITY,
1,025  C   ABSORED POWER, VOLUME ELEMENTS ETC.  ON THE AXIS.
1,026  C
1,027         ZDZV   = ZDZL
1,028         IF (JIN .EQ. 1 .OR. JIN .EQ. IREPZ1) ZDZV = 0.5*ZDZL
1,029         ZDVOL  = ZDZV*ZRDR(1)
1,030         ZDN0DZ = (XNE(1,JZ+1)-XNE(1,JZ))/ZDZC
1,031  C
1,032  C   CALCULATION OF LOCAL GROUP VELOCITY (FIXED GE 0.1C)
1,033  C
1,034         ZVG0       = CLIGHT*SQRT(1.0-ZNE0S(1)*XNC1R)
1,035         ZK0        = ZK*ZVG0/CLIGHT
1,036         ZK0Q       = ZK0*ZK0
1,037         ZKQ        = ZK*ZK
1,038         ZA1        = (ZNES(1)*XNC1R)/ZVG0
1,039         ZA2        = CLIGHQ/(ZVG0*ZVG0*XNC1)
1,040         ZQ0        = CMPLX(0.,-ZK0*ZABSS(1))
```

```
1,041         ZC1       = CMPLX(0.,1./(2.*ZK0))
1,042         ZDEDZ     = ZC1*(ZINC*ZM(1) - ZQ0*ZCFLUX(1))
1,043         IAZ       = 0
1,044         IF(JIN .GT. IREPZ1/2) IAZ=1
1,045         IZZ       = JZ + IAZ
1,046         ZSIGZ2    = 0.
1,047         ZDS2DZ    = 0.
1,048         DABSLA(1) = ZDVOL*ZABSS(1)*ZFLUX(1)
1,049         PWRAB2    = DABSLA(1)
1,050   C
1,051   C   END OF DEFINITION OF CONSTANTS
1,052   C
1,053         IF(NLEMS) THEN
1,054   C
1,055   C   IF THE PONDEROMOTIVE FORCE IS SWITCHED ON THEN
1,056   C   CALCULATION OF THE ON-AXIS PONDEROMOTIVE FORCE
1,057   C
1,058           DEMSR(1) = 0.
1,059           DEMSZ(1) = -ZDVOL*ZA1*(REAL(CONJG(ZCFLUX(1))*ZDEDZ))
1,060          IF(ZVG0 .GT. CTENTH) THEN
1,061           DEMSZ(1) = DEMSZ(1)-0.25*ZDVOL*ZA1*ZA2*ZDN0DZ*ZFLUX(1)
1,062          END IF
1,063         END IF
1,064   C
1,065   C   ZERO ALL COEFFICIENT FROM I4SIG TO IMESHX
1,066   C
1,067           IF (I4SIG .NE. IMESHX) THEN
1,068                DO 754 IB = I4SGP1, IMESHX
1,069                  ZCFLUX(IB) = (0.,0.)
1,070                  ZFLUX(IB)  = 0.
1,071                  DABSLA(IB) = 0.
1,072                  DEMSR(IB)  = 0.
1,073                  DEMSZ(IB)  = 0.
1,074                  ZABSS(IB)  = 0.
1,075                  ZCONJS(IB) = (0.,0.)
1,076                  ZDEDZS(IB) = (0.,0.)
1,077                  ZDEDRS(IB) = (0.,0.)
1,078    754          CONTINUE
1,079           END IF
1,080   C
1,081   C   CALCULATION OF THE ELECTRON DENSITY IN THE CURRENT Z PLANE.
1,082   C   RESTRICTION IS THAT IT CANNOT EXCEED 0.99*CRITICAL DENSITY.
1,083   C
1,084         DO 760 IB=2,I4SIG
1,085           ZNERS(IB) = XNC1R*ZNES(IB)
1,086           ZNRMS(IB) = AMIN1(0.99,ZNERS(IB))
1,087    760  CONTINUE
1,088   C
1,089   C   PFORCE IS A SWITCH PASSED FROM HYDRO CODE.
1,090   C   IF PFORCE > 0.5, THEN LOCAL GROUP VELOCITY IS USED.
1,091   C   IF PFORCE < 0.5, THEN ON AXIS GROUP VELOCITY IS EMPLOYED.
1,092   C   ZDNDZS IS THE GRADIENT OF DENSITY WITH RESPECT TO Z.
```

```
1,093  C
1,094        IF(PFORCE .GT. 0.5) THEN
1,095  C
1,096           DO 762 IB=2,I4SIG
1,097             ZVGS(IB) = CLIGHT*SQRT(1.0-ZNRMS(IB))
1,098   762     CONTINUE
1,099  C
1,100           DO 764 JR=1,JRS(I4SIG)
1,101             ZDNDZS(JR) = (XNE(JR,JZ+1)-XNE(JR,JZ))*ZDZCR
1,102   764     CONTINUE
1,103  C
1,104        ELSE
1,105  C
1,106           DO 766 IB=2,I4SIG
1,107             ZVGS(IB) = ZVG0
1,108   766     CONTINUE
1,109  C
1,110           DO 768 JR=1,JRS(I4SIG)
1,111             ZDNDZS(JR) = ZDN0DZ
1,112   768     CONTINUE
1,113  C
1,114        END IF
1,115  C
1,116  C   COMPUTE THE VOLUME ELEMENTS IN THE CURRENT LASER PLANE
1,117  C
1,118        DO 770 IB=2,IMESHX
1,119          ZAS(IB)    = ZNERS(IB)/ZVGS(IB)
1,120          ZDVOLS(IB) = ZDZV*ZRDR(IB)
1,121   770  CONTINUE
1,122  C
1,123  C  COMPUTE THE TOTAL BEAM AREA ZSIGZ2
1,124  C
1,125        DO 776 IB=2,I4SIG
1,126          ZCONJS(IB) = CONJG(ZCFLUX(IB))
1,127          TEMPS(IB)  = ZRQS(IB)*ZRDR(IB)
1,128          ZSIGZ2     = ZSIGZ2 + TEMPS(IB)*ZFLUX(IB)
1,129   776  CONTINUE
1,130  C
1,131        IF(JIN .EQ. IREPZ1) THEN
1,132  C
1,133          DO 784 IB=2,I4SIG
1,134            ZIMAGS(IB) = AIMAG(ZCFLUX(IB)*CONJG(ZM(IB)))
1,135            ZDS2DZ     = ZDS2DZ + TEMPS(IB)*ZIMAGS(IB)
1,136   784    CONTINUE
1,137  C
1,138        END IF
1,139  C
1,140  C   CALCULATE POWER ABSORPTION FOR EACH CELL (DABSLA) AND
1,141  C   TOTAL POWER ABSORBED (PWRAB2) AT THE CURRENT Z PLANE
1,142  C   CALCULATE POWER OUTPUT FOR AT THE CURRENT Z PLANE.
1,143  C   THIS POWER OUTPUT DOES NOT INCLUED PONDEROMOTIVE WORK
1,144  C
```

```
1,145           PWROUT      = ZRDR(1)*ZFLUX(1)
1,146         DO 790 IB=2,IMESHX
1,147           DABSLA(IB) = ZDVOLS(IB) * ZABSS(IB) * ZFLUX(IB)
1,148           PWRAB2      = PWRAB2 + DABSLA(IB)
1,149           PWROUT      = PWROUT + ZRDR(IB)*ZFLUX(IB)
1,150    790  CONTINUE
1,151           PWRAB2      = ZTWOPI*PWRAB2
1,152           PWROUT      = ZTWOPI*PWROUT
1,153   C
1,154         IF(NLEMS) THEN
1,155   C
1,156   C    IF PONDREROMOTIVE FORCE IS SWITCHED ON,
1,157   C    CALCULATE PONDEROMOTIVE FORCES DEMSZ AND DEMSR
1,158   C    CORRESPOND TO AXIAL AND RADIAL PM FORCES RESPECTIVELY
1,159   C
1,160   C    CALCULATE D/DZ(ZCFLUX) AND D/DR(ZCFLUX)
1,161   C    CORRESPOND TO ZDEDZS AND ZDEDRS RESPECTIVELY.
1,162   C
1,163         DO 792 IB=2,IMESHX
1,164           ZKR2S       = ZKQ*(1.0-ZNRMS(IB))
1,165           ZQRZS       = CMPLX(ZK0Q-ZKR2S,-ZK0*ZABSS(IB))
1,166           ZDEDZS(IB) = ZC1*(ZINC*ZM(IB)-ZQRZS*ZCFLUX(IB))
1,167           ZDEDRS(IB) = ZM(IB-1)*(ZBETA(IB)*ZRRS(IB)+1.0/6.0
1,168        1               * ZRQS(IB)*ZDRRS(IB))+ZM(IB)*(ZALFA(IB)
1,169        2               * ZRRS(IB)+ZDRRS(IB)*ZR(IB)*(1.0/3.0*ZR(IB)
1,170        3               - 0.5*ZR(IB-1)))+ZRRS(IB)*ZGAMA(IB)
1,171        4               * (ZCFLUX(IB)-ZCFLUX(IB-1))
1,172    792   CONTINUE
1,173   C
1,174         DO 800 IB=2,IMESHX
1,175           DEMSZ(IB) = -ZDVOLS(IB)*ZAS(IB)*REAL(ZCONJS(IB)
1,176        &              * ZDEDZS(IB))
1,177    800   CONTINUE
1,178   C
1,179           DO 825 IB=2,IMESHX
1,180   C
1,181             IF(ZVGS(IB) .GT. CTENTH) THEN
1,182               DEMSZ(IB) = DEMSZ(IB)-0.25*ZDVOLS(IB)*ZAS(IB)
1,183        1                  * CQXR/(ZVGS(IB)*ZVGS(IB))
1,184        2                  * ZDNDZS(JRS(IB))*ZFLUX(IB)
1,185             END IF
1,186   C
1,187    825    CONTINUE
1,188   C
1,189           DO 830 IB=2,IMESHX
1,190             DEMSR(IB) = 0.0
1,191    830    CONTINUE
1,192           DO 835 IB=2,IMESHX
1,193             IF(IB .GT. (1+(IREPR(JRS(IB))-1)/2)) THEN
1,194               DEMSR(IB) = -ZDVOLS(IB)*ZAS(IB)
1,195        &                  * REAL(ZCONJS(IB)*ZDEDRS(IB))
1,196             END IF
```

```
1,197   835    CONTINUE
1,198  C
1,199        END IF
1,200  C
1,201  C -------- EXIT IF POWER DOWN TO 0.1% OF INPUT POWER --------
1,202  C
1,203           IF ((FCPI*ZNORM) .LT. (.001*ZPL1)) GO TO 900
1,204  C
1,205  C
1,206  C    DEFINE 4-SIGMA INDEX ACCORDING TO SPATIAL PULSE SHAPE.
1,207  C
1,208           IF (ISHAPE .EQ. 1) THEN
1,209             ZSIGZ2 = ZSIGZ2 / ZNORM
1,210             ZDS2DZ = ZDS2DZ / (ZNORM*ZK0)
1,211             ZR4SIG = 4. * SQRT(ZSIGZ2)
1,212  C
1,213  C    FIND NEW 4-SIGMA INDEX AND DO NOT REDEFINE MAXIMUM OF
1,214  C    LASER GRID POINTS WHEN ON A MHD GRID POINT.
1,215  C
1,216           IF (JIN .NE. IREPZ1) THEN
1,217  C
1,218              DO 840 JL = 1, IMESHX
1,219               IF (ZR(JL) .LE. ZR4SIG) GO TO 840
1,220                I4SIG  = JL
1,221                I4SGM1 = I4SIG - 1
1,222                I4SGP1 = I4SIG + 1
1,223                GO TO 842
1,224     840       CONTINUE
1,225               END IF
1,226     842      CONTINUE
1,227  C
1,228  C    LOOK FOR SMALLEST AND LARGEST SIGMAS
1,229  C
1,230           IF (ZSIGZ2 .LT. ZSIG1**2) ZSIG1 = SQRT(ZSIGZ2)
1,231           IF (ZSIGZ2 .GT. BRL1**2) BRL1 = SQRT(ZSIGZ2)
1,232  C
1,233  C    BRANCH HERE IF PLANE WAVE + GAUSSIAN IS SELECTED.
1,234  C    I.E., DO NOT REDEFINE ZR4SIG FOR THIS PULSE SHAPE.
1,235  C
1,236         END IF
1,237  C
1,238  C ---------------------------------------------------------------
1,239  C                  ENSURE ENERGY CONSERVATION
1,240  C ---------------------------------------------------------------
1,241           IF (NLEMS) THEN
1,242  C
1,243  C    ZVELRS AND ZVELZS ARE THE FLUID VELOCITY IN RADIAL
1,244  C    AND IN AXIAL RESPECTIVELY
1,245  C
1,246              DO 850 JL=1,IMESHX
1,247                ZVELRS(JL) = ZRHODS(JL)*
1,248       1          ( ZETAD*RHOVR(JRS(JL),JZ)/RHO(JRS(JL),JZ)
```

```
1,249        2          + ZETA*RHOVR(JRS(JL),JZP1)/RHO(JRS(JL),JZP1) )
1,250        3          + ZRHOS(JL)*
1,251        4          ( ZETAD*RHOVR(JRP1S(JL),JZ)/RHO(JRP1S(JL),JZ)
1,252        5          + ZETA*RHOVR(JRP1S(JL),JZP1)/RHO(JRP1S(JL),JZP1) )
1,253                  ZVELZS(JL) = ZRHODS(JL)*
1,254        1          ( ZETAD*RHOVZ(JRS(JL),JZ)/RHO(JRS(JL),JZ)
1,255        2          + ZETA*RHOVZ(JRS(JL),JZP1)/RHO(JRS(JL),JZP1) )
1,256        3          + ZRHOS(JL)*
1,257        4          ( ZETAD*RHOVZ(JRP1S(JL),JZ)/RHO(JRP1S(JL),JZ)
1,258        5          + ZETA*RHOVZ(JRP1S(JL),JZP1)/RHO(JRP1S(JL),JZP1) )
1,259     850     CONTINUE
1,260  C
1,261  C    CALCULATE WORK DONE BY THE PONDEROMOTIVE FORCES IN R
1,262  C    AND IN Z. PWRPO2 IS TOTAL WORK DONE BY THESE FORCES
1,263  C    AT THE CURRENT Z PLANE
1,264  C
1,265               PWRPO2 = 0.0
1,266               DO 852 JL=1,IMESHX
1,267                 WORKR(JL) = DEMSR(JL)*ZVELRS(JL)
1,268                 WORKZ(JL) = DEMSZ(JL)*ZVELZS(JL)
1,269                 PWRPO2    = PWRPO2 + ( WORKR(JL) + WORKZ(JL) )
1,270     852       CONTINUE
1,271               PWRPO2    = ZTWOPI*PWRPO2
1,272           END IF
1,273  C
1,274  C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1,275  C    SCALE THE BEAM WHEN PONDEROMOTIVE FORCE IS PRESENT
1,276  C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1,277  C
1,278  C    ENERGY REMAINIG = ENERGY AVAL. -  WORK DONE
1,279  C    SCALING FACTOR ZZ ACCOUNTS FOR ENTIRE BEAM TO BE SCALED
1,280  C    BY A CONSTANT FACTOR. THIS IS USED TO ACCOUNT FOR WORKDONE
1,281  C    BY PONDEROMOTIVE FORCES.
1,282  C
1,283           ZZ = 1.0
1,284           IF (JIN .NE. 1) THEN
1,285             IF (JIN .NE. IREPZ1) THEN
1,286              PWRAB2 = 0.5*PWRAB2
1,287              PWRPO2 = 0.5*PWRPO2
1,288             ENDIF
1,289  C
1,290  C    PWRABS IS THE TOTAL ABSORPTION POWER FROM PREVIOUS PLANE
1,291  C    PLUS CURRENT PLANE. PWRPON IS THE TOTAL WORK DONE BY THE
1,292  C    PONDEROMOTIVE FORCES FROM PREVIOUS PLANE PLUS CURRENT PLANE
1,293  C
1,294            PWRABS = PWRAB1 + PWRAB2
1,295            PWRPON = PWRPO1 + PWRPO2
1,296  C
1,297  C    CALCULATE SCALING FACTOR ZZ
1,298  C
1,299           ZZ = PWRIN / (PWROUT + PWRABS + PWRPON)
1,300  C
```

```
1,301  C   RESCALE DABSLA, ZFLUX, DEMSR & DEMSZ TO ACCOUNT FOR
1,302  C   PM WORK ON PLASMA
1,303  C
1,304         DO 855 JL = 1, IMESHX
1,305          DABSLA(JL) = ZZ * DABSLA(JL)
1,306          ZFLUX(JL)  = ZZ * ZFLUX(JL)
1,307   855   CONTINUE
1,308  C
1,309         IF (NLEMS) THEN
1,310         DO 860 JL = 1,IMESHX
1,311          DEMSR(JL) = ZZ * DEMSR(JL)
1,312          DEMSZ(JL) = ZZ * DEMSZ(JL)
1,313   860   CONTINUE
1,314         ENDIF
1,315  C
1,316  C   ZPOND AND ZPABS ARE CUMULATIVE PONDEROMOTIVE WORKS AND
1,317  C   ABSORBED POWER FROM THE ENTRANCE PLASMA UP TO CURRENT
1,318  C   Z PLANE, RESPECTIVELY.
1,319  C
1,320         PWRIN = ZZ * PWROUT
1,321         ZPOND = ZPOND + ZZ * PWRPON
1,322         ZPABS = ZPABS + ZZ * PWRABS
1,323  C
1,324         ENDIF
1,325         PWRAB1 = ZZ * PWRAB2
1,326         PWRPO1 = ZZ * PWRPO2
1,327  C
1,328  C-----------------------------------------------------------------
1,329  C   ADD THE SCALED CONTRIBUTIONS TO THEIR RESPECTIVE ARRAYS
1,330  C   ASSIGN TO HYDRO GRID
1,331  C
1,332          DO 870 IL=1,IMESH
1,333           ABSLAS(IRRS(IL),IZZ) = ABSLAS(IRRS(IL),IZZ)+DABSLA(IL)
1,334   870    CONTINUE
1,335  C
1,336        IF(NLEMS) THEN
1,337  C
1,338           DO 875 IL=1,IMESH
1,339             EMSR(IRRS(IL),IZZ) = EMSR(IRRS(IL),IZZ)+DEMSR(IL)
1,340             EMSZ(IRRS(IL),IZZ) = EMSZ(IRRS(IL),IZZ)+DEMSZ(IL)
1,341   875     CONTINUE
1,342  C
1,343        END IF
1,344  C
1,345  C   ASSIGN THE FLUX TO THE HYDRO GRID IF ON MHD Z-PLANE
1,346  C
1,347           IF (JIN .EQ. IREPZ1) THEN
1,348  C
1,349             DO 880 JR=1,NR
1,350               TGR = 0.
1,351               DO 882 J=JMINS(JR),JMAXS(JR)
1,352                 TGR = TGR + ZRDR(J)*ZFLUX(J)
```

```
 882          CONTINUE
              FLUX(JR,JZP1) = TGR/RDR(JR)
 880        CONTINUE
C
          END IF
C
C   STATEMENT FUNCTION FOR LANGDON CORRECTION FACTOR:-
C   FLANGD(AL) = 1.-0.553*AL/(0.375+AL)
C-----------------------------------------------------------
C   UPDATE THE LANGDON CORRECTION FACTOR
C
      DO 890 JL=1,IMESHX
       ZARG       = (XL1QF*ZAZS(JL)*ZFLUX(JL)
     &             / (ZVG0S(JL)*ZTES(JL)))**.75
       WLANGD(JL) = 1.0-0.553*ZARG/(0.375+ZARG)
 890  CONTINUE
C-----------------------------------------------------------
 720   CONTINUE
C --------------- END OF INNER Z-LOOP ----------------------
 710  CONTINUE
C -------------   END OF OUTER Z-LOOP ----------------------
C   END BLOCK 6
C-----------------------------------------------------------
 900    CONTINUE
C
C  BLOCK 7     NORMALIZE BY VOLUME ELEMENTS
C
      DO 910 JR = 1, NR
        DO 910 JZ = 1, NZ
          IZ1           = MAX0(1,JZ - 1)
          IZ2           = MIN0(NZ,JZ + 1)
          ZDZV          = 0.5 * (CZ(IZ2) - CZ(IZ1))
          ZINV          = 1. / (RDR(JR)*ZDZV)
          ABSLAS(JR,JZ) = ZINV * ABSLAS(JR,JZ)
          IF (NLEMS) THEN
            EMSR(JR,JZ) = ZINV * EMSR(JR,JZ)
            EMSZ(JR,JZ) = ZINV * EMSZ(JR,JZ)
          END IF
 910  CONTINUE
C
C-----------------------------------------------------------
C   COMPUTE LASER POWER OUTPUT FROM PLASMA
C
         TGR = 0.0
         DO 920 J=1,NR
          TGR = TGR + RDR(J)*FLUX(J,NZ)
 920     CONTINUE
         ZPOUT = ZTWOPI*TGR
C
C-----------------------------------------------------------
C   SET VALUES OF ARRAYS AT NR EQUAL TO THAT AT NRM1.
C   THIS IS CONSISTENT WITH BOUNDARY CONDITIONS IN CASTOR.
```

```
C
      DO 930 JZ = 1, NZ
         FLUX(NR,JZ)    = FLUX(NRM1,JZ)
         ABSLAS(NR,JZ) = ABSLAS(NRM1,JZ)
       IF (NLEMS) THEN
         EMSR(NR,JZ) = EMSR(NRM1,JZ)
         EMSZ(NR,JZ) = EMSZ(NRM1,JZ)
       END IF
 930   CONTINUE
C
       BFLUX = (ZPL1 - ZPOUT - ZPOND - ZPABS) / ZPL1

C
C   END BLOCK 7
C
      RETURN
      END
```

```
      SUBROUTINE CYCLIC(N, TRM, Y, XX)
C
C        SOLVE TRIDIAGONAL MATRIX EQUATION M Z = Y
C              B(I-1)Z(I-1) + A(I)Z(I) + C(I)Z(I+1) = Y(I)
C              SOLVE FOR Z(I), I=1,2,.....,N
C              ASSUME B(0) = C(N) = 0
C
C   N    =  ORDER OF THE MATRIX FOR SOLUTION
C
C   TRM  =  TRIDIAGONAL MATRIX STORED BY COLUMNS :-
C
C              1ST COLUMN (STARTS ROW2) CONTAINS LOWER DIAGONAL
C              2ND COLUMN (STARTS ROW1) CONTAINS CENTRE DIAGONAL
C              3RD COLUMN (STARTS ROW1) CONTAINS UPPER DIAGONAL
C
C   XX   =  SOLUTION VECTOR
C   Y    =  RIGHT HAND SIDE VECTOR ELEMENTS
C   RQ   =  VECTOR OF R-VALUES SUCH THAT R*2**Q .LE. N
C
C-----------------------------------------------------------------
C Q IS THE LEVEL OF REDUCTION.
C Q VARIES AS 0,1,.....,P WHERE
C P IS THE HIGHEST POWER OF 2 WITH 2**P <= N
C-----------------------------------------------------------------
      PARAMETER (MAXP=9,NFMMAX=300)
C
C MAXP IS THE MAXIMUM NUMBER OF PERMITTED REDUCTIONS
C MAXP = 9 CORRESPONDS TO A MAXIMUM VECTOR LENGTH OF 512
C NFMMAX IS THE MAXIMUM MATRIX DIMENSION PERMITTED
C AND IS DIMENSIONED ACCORDING TO HEATER VARIABLES.
C
      COMPLEX TRM(NFMMAX,3), Y(NFMMAX), XX(NFMMAX)
C
C LOCAL VARIABLES TO AID VECTORIZATION
C
      COMPLEX X(NFMMAX,0:MAXP),  L(NFMMAX,0:MAXP),
     1        U(NFMMAX,0:MAXP),  D(NFMMAX,0:MAXP),
     2        DD(NFMMAX,0:MAXP), UU(NFMMAX,0:MAXP),
     3        LL(NFMMAX,0:MAXP), W(NFMMAX,0:MAXP)
C
      INTEGER Q, P, R, RM1, RP1, R2, R2M1, RQ(0:MAXP)
C
C FIND P (NUMBER OF REDUCTION LEVELS)
C
      P = ALOG(FLOAT(N)) / ALOG(2.0)
      IF (2.**P .GT. N) P=P-1
C
C CALCULATE R(Q,N) SUCH THAT R(Q,N) * 2**Q .LE. N
C R(Q,N) IS THE REDUCTION LEVEL
C
      DO 10 Q = 0, P
        RQ(Q) = N / (2 ** Q)
```

```
 10 CONTINUE
C
C CLEAR ALL PRIOR CONDITIONS TO ENSURE CORRECT CALCULATIONS
C
      DO 20 I = 0, P+1
      DO 20 J = 1, N+1
        L(J,I)  = (0.,0.)
        LL(J,I) = (0.,0.)
        U(J,I)  = (0.,0.)
        UU(J,I) = (0.,0.)
        D(J,I)  = (0.,0.)
        DD(J,I) = (0.,0.)
        W(J,I)  = (0.,0.)
        X(J,I)  = (0.,0.)
 20 CONTINUE
C
C INITIALIZATION VECTOR ELEMENTS (COPY DIAGONAL ELEMENTS)
C ASSIGN THE UPPER, LOWER AND DIAGONAL ELEMENTS
C
      DO 40 I = 1, N-1
       L(I,0) = TRM(I+1,1)
       U(I,0) = TRM(I,3)
 40 CONTINUE
C
      DO 50 I = 1, N
       D(I,0) = TRM(I,2)
       W(I,0) = Y(I)
 50 CONTINUE
C
C LU DECOMPOSITION (Q DENOTES CURRENT LEVEL OF REDUCTION)
C
      DO 140 Q = 0, P-2
C
        R = RQ(Q)
       RM1 = R - 1
       RP1 = R + 1
C
       DO 60 I = 2, RP1, 2
         DD(I-1,Q) = 1.0 / D(I-1,Q)
 60    CONTINUE
C
       DO 70 I = 2, R, 2
         LL(I-1,Q) = L(I-1,Q) * DD(I-1,Q)
         UU(I-1,Q) = U(I-1,Q)
 70    CONTINUE
C
       DO 80 I = 2, RM1, 2
         LL(I,Q) = L(I,Q)
         UU(I,Q) = U(I,Q) * DD(I+1,Q)
 80    CONTINUE
C
         R2 = RQ(Q+1)
```

```
105           R2M1 = R2 - 1
106   C
107           DO 110 I = 1, R2
108             D(I,Q+1) = D(I*2,Q) - LL(I*2-1,Q) * UU(2*I-1,Q)
109          1                    - LL(I*2,Q) * UU(I*2,Q)
110     110   CONTINUE
111           DO 120 I = 1, R2M1
112             L(I,Q+1) = - LL(I*2,Q) * LL(I*2+1,Q)
113             U(I,Q+1) = - UU(I*2,Q) * UU(I*2+1,Q)
114     120   CONTINUE
115   C
116     140 CONTINUE
117   C
118   C THE LAST TWO LEVELS OF REDUCTION : Q = P-1, P
119   C
120           R = RQ(P-1)
121         RM1 = R - 1
122         RP1 = R + 1
123          R2 = 1
124
125         DO 150 I = 2, RP1, 2
126           DD(I-1,P-1) = 1.0 / D(I-1,P-1)
127     150 CONTINUE
128   C
129         DO 160 I = 2, R, 2
130           LL(I-1,P-1) = L(I-1,P-1) * DD(I-1,P-1)
131           UU(I-1,P-1) = U(I-1,P-1)
132     160 CONTINUE
133   C
134         DO 170 I = 2, RM1, 2
135           LL(I,P-1) = L(I,P-1)
136           UU(I,P-1) = U(I,P-1) * DD(I+1,P-1)
137     170 CONTINUE
138   C
139         D(1,P) = D(2,P-1) - LL(1,P-1) * UU(1,P-1)
140        1         - LL(2,P-1) * UU(2,P-1)
141         DD(1,P) = 1.0 / D(1,P)
142   C
143   C CALCULATE W VECTOR FROM Q = 0,1,...,P-2,P-1
144   C AND FROM I = 1,2,....,RQ(Q+1)
145   C
146         DO 210 Q = 0, P-1
147           DO 200 I = 1, RQ(Q+1)
148             W(I,Q+1) = W(I*2,Q) - LL(2*I-1,Q) * W(2*I-1,Q)
149        1                    - UU(2*I,Q) * W(2*I+1,Q)
150     200   CONTINUE
151     210 CONTINUE
152   C
153   C-------------------------------------------------------------
154   C SOLVE FOR X USING BACK SUBSTITUTION ( LOOP FROM
155   C              Q = (P-1), (P-2),....2, 1, 0 )
156   C
```

```
157         X(1,P) = DD(1,P) * W(1,P)
158         DO 240 Q = P-1, 0, -1
159   C
160             R = RQ(Q)
161           RP1 = R + 1
162   C
163   C EVEN TERMS
164   C
165           DO 220 I = 1, R/2
166             X(2*I,Q) = X(I,Q+1)
167     220   CONTINUE
168   C
169   C ODD TERMS
170   C
171           DO 230 I = 2, RP1, 2
172             X(I-1,Q) = DD(I-1,Q) * ( W(I-1,Q) - LL(I-2,Q)
173        1                * X(I-2,Q) - UU(I-1,Q) * X(I,Q) )
174     230   CONTINUE
175     240 CONTINUE
176   C
177   C THE SOLUTION VECTOR
178   C
179         DO 250 I = 1, N
180           XX(I) = X(I,0)
181     250 CONTINUE
182   C
183         RETURN
184         END
```